Application of Stochastic Learning Automata to Intelligent Vehicle Control

FLORIN STOICA, EMIL M. POPA
Computer Science Department
University "Lucian Blaga" Sibiu
Str. Dr. Ion Ratiu 5-7, 550012, Sibiu
ROMANIA

Abstract: - A stochastic automaton can perform a finite number of actions in a random environment. When a specific action is performed, the environment responds by producing an environment output that is stochastically related to the action. This response may be favourable or unfavourable. The aim is to design an automaton that can determine the best action guided by past actions and responses. Using Stochastic Learning Automata techniques, we introduce a decision/control method for intelligent vehicles, in an infrastructure managed architecture. The aim is to design an automata system that can learn the best possible action based on the data received from on-board sensors or from the localization system of highway infrastructure.

Key-Words: - Stochastic Learning Automata, Reinforcement Learning, Intelligent Vehicle Control

1 Introduction

An *automaton* is a machine or control mechanism designed to automatically follow a predetermined sequence of operations or respond to encoded instructions. The term *stochastic* emphasizes the adaptive nature of the automaton we describe here. The automaton described here does not follow predetermined rules, but adapts to changes in its environment. This adaptation is the result of the *learning* process. Learning is defined as any permanent change in behavior as a result of past experience, and a learning system should therefore have the ability to improve its behavior with time, toward a final goal.

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). 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*. The algorithm that guarantees the desired learning process is called a *reinforcement scheme* [5].

Mathematically, the environment is defined by a triple $\{\alpha, c, \beta\}$ where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents a finite set of actions being the input to the environment, $\beta = \{\beta_1, \beta_2\}$ represents a binary response set, and $c = \{c_1, c_2, ..., c_r\}$ is a set of penalty probabilities, where c_i is the probability that action α_i will result in an unfavourable response. Given that $\beta(n) = 0$ is a favourable outcome and $\beta(n) = 1$ is an unfavourable outcome at time instant $n\,(n = 0, 1, 2, ...)$, the element c_i of c is defined mathematically by:

$c_i = P(\beta(n) = 1 \mid \alpha(n) = \alpha_i) \quad i = 1, 2, ..., r$

The response values can be represented in three different models. In the P-model (described above), the response values are either 0 or 1, in the S-model the response values is continuous in the range (0, 1) and in the Q-model the values is in a finite set of discrete values in the range (0, 1).

The environment can further be split up in two types, stationary and nonstationary. In a stationary environment the penalty probabilities will never change. In a nonstationary environment the penalties will change over time.

2 Formal definition of a Stochastic Learning Automaton

The automaton is defined by a quintuple $\{\Phi, \alpha, \beta, F(\bullet,\bullet), H(\bullet,\bullet)\}$.

$\Phi = \{\Phi_1, \Phi_2, ..., \Phi_s\}$ is the set of internal states of the automaton. The internal states determine the action to be produced by the automaton. The state at time instant n is denoted by $\Phi(n)$ and is an element of the finite set Φ.

$\alpha = \{\alpha_1, \alpha_{2,} ..., \alpha_r\}$ denotes the set of actions that can be produced by the automaton. This is the output set of the automaton, hence also being the input set to the environment. The action done at time instant n is denoted $\alpha(n)$ and is an element of the finite set α.

$\beta = \{\beta_1, \beta_2, ..., \beta_m\}$ or $\beta = \{(a, b)\}$ is the input set to the automaton, that is the set of responses from the environment. This set can be either finite or infinite. $\beta(n)$ denotes the input to the automaton at time instant n.

$F(\bullet,\bullet)$ is a function that maps the current state and the response from the environment into the next state, given mathematically by $F(\bullet,\bullet):\Phi\times\beta\to\Phi$. This formula is called the transition function. A similar expression is showed by the formula $\Phi(n+1)=F[\Phi(n),\beta(n)]$.

The transition function can be either deterministic or stochastic. If the function is deterministic the result of the function is uniquely specified for each state.

If the transition function F is stochastic, the elements f_{ij}^{β} of F represent the probability that the automaton moves from state Φ_i to state Φ_j given the input β:

$f_{ij}^{\beta}=P(\Phi(n+1)=\Phi_j \mid \Phi(n)=\Phi_i,\beta(n)=\beta)$

$i,j=1,2,...,s$ and $\beta\in\{\beta_1,\beta_2,...,\beta_m\}$

$H(\bullet,\bullet)$ is the output function and is defined mathematically by $H(\bullet,\bullet):\Phi\times\beta\to\alpha$. This function maps the current state and the response from the environment into the action produced by the automaton. If the current output depends on only the current state, the automaton is referred to as *state-output automaton*. In this case the function $H(\bullet,\bullet)$ is replaced by an output function $G(\bullet):\Phi\to\alpha$ which can be either deterministic or stochastic: $\alpha(n)=G[\Phi(n)]$.

If G is stochastic, the elements of this set are denoted g_{ij}. The value of this element represents the probability that the action done by the automaton is α_j given the automaton is in state Φ_i:

$g_{ij}=P(\alpha(n)=\alpha_j \mid \Phi(n)=\Phi_i)\ \ i=\overline{1,s}\ \ j=\overline{1,r}$

In short the automaton takes an input from the environment and produces an action based on this. The automaton is showed in figure 1:

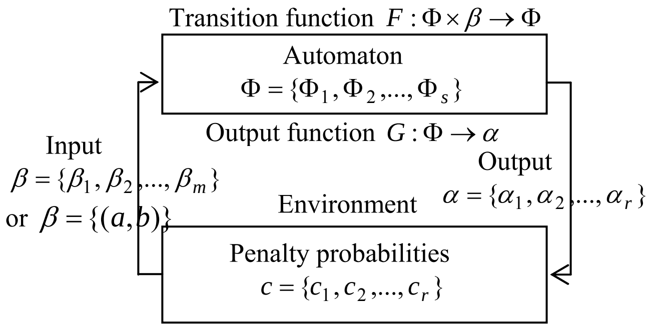
Transition function $F:\Phi\times\beta\to\Phi$
Automaton
$\Phi=\{\Phi_1,\Phi_2,...,\Phi_s\}$
Output function $G:\Phi\to\alpha$
Input
$\beta=\{\beta_1,\beta_2,...,\beta_m\}$
or $\beta=\{(a,b)\}$
Output
$\alpha=\{\alpha_1,\alpha_2,...,\alpha_r\}$
Environment
Penalty probabilities
$c=\{c_1,c_2,...,c_r\}$

Fig. 1 A stochastic automaton

An automaton is called to be fixed structured when the functions f_{ij} and g_{ij} are having values that do not change over time. By making them change over time, one can get a greater flexibility where actions rewarded will get a higher chance of being chosen again. Such an automaton is called *variable-structure automaton*. Furthermore, in the case of variable-structure automaton, the above definitions of the transition functions F and G are not used explicitly. In order to describe the reinforcement schemes, is defined $p(n)$, a vector of action probabilities: $p_i(n)=P(\alpha(n)=\alpha_i),\ i=\overline{1,r}$

Updating action probabilities can be represented as follows:

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

where T is a mapping. This formula says the next action probability $p(n+1)$ is updated based on the current probability $p(n)$, the input from the environment and the resulting action. If $p(n+1)$ is a linear function of $p(n)$, the reinforcement scheme is said to be linear; otherwise it is termed nonlinear.

3 Variable Structure Automaton

3.1 Performance Evaluation

A learning automaton generates a sequence of actions on the basis of its interaction with the environment. If the automaton is “learning” in the process, its performance must be superior to “intuitive” methods. In the following we will consider the simplest case, the P-model and stationary random environments.

Consider a stationary random environment with penalty probabilities

$\{c_1,c_2,...,c_r\}$ where $c_i=P(\beta(n)=1 \mid \alpha(n)=\alpha_i)$.

We define a quantity $M(n)$ as the average penalty for a given action probability vector:

$M(n)=P(\beta(n)=1 \mid p(n))=$

$$\sum_{i=1}^{r}P(\beta(n)=1 \mid \alpha(n)=\alpha_i)*P(\alpha(n)=\alpha_i)=\sum_{i=1}^{r}c_i p_i(n)$$

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*: $M(n+1)<M(n)$ for all n [8].

Absolutely expedient learning schemes are presently the only class of schemes for which necessary and sufficient conditions of design are available. The algorithm we will present in this paper is derived from a nonlinear absolutely expedient reinforcement scheme presented by [7].

3.2 Absolutely expedient reinforcement schemes

The reinforcement scheme is the basis of the learning

process for learning automata. The general solution for absolutely expedient schemes was found by Lakshmivarahan and Thathachar [10].
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”). In a stationary N-teacher P-model environment, if an automaton produced the action α_i and the environment responses are β_i^j $j = 1,...,N$ at time instant n, then the vector of action probabilities $p(n)$ is updated as follows [7]:

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

$$-\left[1-\frac{1}{N}\sum_{k=1}^{N}\beta_i^k\right] * \sum_{\substack{j=1\\ j\neq i}}^{r}\psi_j(p(n)) \qquad (1)$$

$$p_j(n+1) = p_j(n) - \left[\frac{1}{N}\sum_{k=1}^{N}\beta_i^k\right] * \phi_j(p(n)) +$$

$$+\left[1-\frac{1}{N}\sum_{k=1}^{N}\beta_i^k\right] * \psi_j(p(n))$$

for all $j \neq i$ where the functions ϕ_i and ψ_i satisfy the following conditions:

$$\frac{\phi_1(p(n))}{p_1(n)} = ... = \frac{\phi_r(p(n))}{p_r(n)} = \lambda(p(n)) \quad (2)$$

$$\frac{\psi_1(p(n))}{p_1(n)} = ... = \frac{\psi_r(p(n))}{p_r(n)} = \mu(p(n))$$

$$p_i(n) + \sum_{\substack{j=1\\ j\neq i}}^{r}\phi_j(p(n)) > 0 \qquad (3)$$

$$p_i(n) - \sum_{\substack{j=1\\ j\neq i}}^{r}\psi_j(p(n)) < 1 \qquad (4)$$

$$p_j(n) + \psi_j(p(n)) > 0 \qquad (5)$$

$$p_j(n) - \phi_j(p(n)) < 1 \qquad (6)$$

for all $j \in \{1,...,r\} \setminus \{i\}$
The conditions (3)-(6) ensure that $0 < p_k < 1,\ k = \overline{1,r}$.

Theorem If the functions $\lambda(p(n))$ and $\mu(p(n))$ satisfy the following conditions:

$$\lambda(p(n)) \le 0$$
$$\mu(p(n)) \le 0 \qquad (7)$$
$$\lambda(p(n)) + \mu(p(n)) < 0$$

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 [9].

4 A new nonlinear reinforcement scheme

Because the above theorem is also valid for a single-teacher model, we can define a single environment response that is a function f of many teacher outputs.
Thus, we can update the above algorithm as follows:

$$p_i(n+1) = p_i(n) + f * (-\theta * H(n)) * [1 - p_i(n)] -$$
$$-(1-f) * (-\theta) * [1 - p_i(n)]$$
$$p_j(n+1) = p_j(n) - f * (-\theta * H(n)) * p_j(n) + \qquad (8)$$
$$+(1-f) * (-\theta) * p_j(n)$$

for all $j \neq i$, i.e.:

$$\psi_k(p(n)) = -\theta * p_k(n)$$
$$\phi_k(p(n)) = -\theta * H(n) * p_k(n)$$

where the learning parameter θ is a real value which satisfy: $0 < \theta < 1$.
The function H is defined as:

$$H(n) = \min\left\{1; \max\left\{\min\left\{\frac{p_i(n)}{\theta(1-p_i(n))} - \varepsilon, \left(\frac{1-p_j(n)}{\theta * p_j(n)} - \varepsilon\right)_{\substack{j=\overline{1,r}\\ j\neq i}}\right\}; 0\right\}\right\}$$

Parameter ε is an arbitrarily small positive real number.
Our reinforcement scheme differs from the one given in [6], [7] by the definition of these two functions: H and ϕ_k.
We now show that are satisfied all the conditions of the reinforcement scheme.
From (2) we have:

$$\frac{\phi_k(p(n))}{p_k(n)} = \frac{-\theta * H(n) * p_k(n)}{p_k(n)} = -\theta * H(n) = \lambda(p(n))$$

$$\frac{\psi_k(p(n))}{p_k(n)} = \frac{-\theta * p_k(n)}{p_k(n)} = -\theta = \mu(p(n))$$

The rest of the conditions translate to the following:
Condition (3):

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

$$p_i(n) - \theta * H(n) * (1 - p_i(n)) > 0 \Leftrightarrow$$

$$\theta * H(n) * (1 - p_i(n)) < p_i(n) \Leftrightarrow H(n) < \frac{p_i(n)}{\theta * (1 - p_i(n))}$$

This condition is satisfied by the definition of the function $H(n)$.
Condition (4):

$$p_i(n) - \sum_{\substack{j=1\\ j\neq i}}^{r}\psi_j(p(n)) < 1 \Leftrightarrow p_i(n) + \theta * (1 - p_i(n)) < 1$$

But $p_i(n) + \theta * (1 - p_i(n)) < p_i(n) + 1 - p_i(n) = 1$ since $0 < \theta < 1$

Condition (5):

$p_j(n) + \psi_j(p(n)) > 0 \Leftrightarrow p_j(n) - \theta * p_j(n) > 0$ for all $j \in \{1,...,r\} \setminus \{i\}$

But $p_j(n) - \theta * p_j(n) = p_j(n) * (1 - \theta) > 0$ since $0 < \theta < 1$ and $0 < p_j(n) < 1$ for all $j \in \{1,...,r\} \setminus \{i\}$

Condition (6):

$p_j(n) - \phi_j(p(n)) < 1 \Leftrightarrow p_j(n) + \theta * H(n) * p_j(n) < 1$ for all $j \in \{1,...,r\} \setminus \{i\}$

We have:

$p_j(n) + \theta * H(n) * p_j(n) < 1 \Leftrightarrow H(n) < \dfrac{1 - p_j(n)}{\theta * p_j(n)}$ for all $j \in \{1,...,r\} \setminus \{i\}$.

This condition is satisfied by the definition of the function $H(n)$.

With all conditions of the equations (1) satisfied, we conclude that the reinforcement scheme is a candidate for absolute expediency.

Furthermore, the functions λ and μ for our nonlinear scheme satisfy the following:

$$\lambda(p(n)) = -\theta * H(n) \le 0$$
$$\mu(p(n)) = -\theta \le 0$$
$$\lambda(p(n)) + \mu(p(n)) = -\theta * (1 + H(n)) < 0$$

because $0 < \theta < 1$ and $0 \le H(n) \le 1$

In conclusion, we state the algorithm given in equations (8) is absolutely expedient in a stationary environment.

5 The model

The task of creating intelligent systems that we can rely on is not trivial. In this section, we present a method for intelligent vehicle control, having as theoretical background Stochastic Learning Automata. We visualize the planning layer of an intelligent vehicle as an automaton (or automata group) in a nonstationary environment. We attempt to find a way to make intelligent decisions here, having as objectives conformance with traffic parameters imposed by the highway infrastructure (management system and global control), and improved safety by minimizing crash risk. The aim here is to design an automata system that can learn the best possible action based on the data received from on-board sensors, of from roadside-to-vehicle communications. For our model, we assume that an intelligent vehicle is capable of two sets of lateral and longitudinal actions. Lateral actions are LEFT (shift to left lane), RIGHT (shift to right lane) and LINE_OK (stay in current lane). Longitudinal actions are ACC (accelerate), DEC (decelerate) and SPEED_OK (keep current speed). An autonomous vehicle must be able to “sense” the environment around itself. Therefore, we assume that there are four different sensors modules on board the vehicle (the headway module, two side modules and a speed module), in order to detect the presence of a vehicle traveling in front of the vehicle or in the immediately adjacent lane and to know the current speed of the vehicle. These sensor modules evaluate the information received from the on-board sensors or from the highway infrastructure in the light of the current automata actions, and send a response to the automata. Our basic model for planning and coordination of lane changing and speed control is shown in figure 2.

The response from physical environment is a combination of outputs from the sensor modules. Because an input parameter for the decision blocks is the action chosen by the stochastic automaton, is necessary to use two distinct functions F_1 and F_2 for mapping the outputs of decision blocks in inputs for the two learning automata, namely the longitudinal automaton and respectively the lateral automaton.

After updating the action probability vectors in both learning automata, using the nonlinear reinforcement scheme presented in section 4, the outputs from stochastic automata are transmitted to the regulation layer. The regulation layer handles the actions received from the two automata in a distinct manner, using for each of them a regulation buffer. If an action received was rewarded, it will be introduced in the regulation buffer of the corresponding automaton, else in buffer will be introduced a certain value which denotes a penalized action by the physical environment. The regulation layer does not carry out the action chosen immediately; instead, it carries out an action only if it is recommended k times consecutively by the automaton, where k is the length of the regulation buffer. After an action is executed, the action probability vector is initialized to $\frac{1}{r}$, where r is the number of actions. When an action is executed, regulation buffer is initialized also.

6 Sensor modules

The four teacher modules mentioned above are decision blocks that calculate the response (reward/penalty), based on the last chosen action of automaton. Table 1 describes the output of decision blocks for side sensors.

As seen in table 1, a penalty response is received from the left sensor module when the action is LEFT and there is a vehicle in the left or the vehicle is already traveling on the leftmost lane. There is a similar situation for the right sensor module.

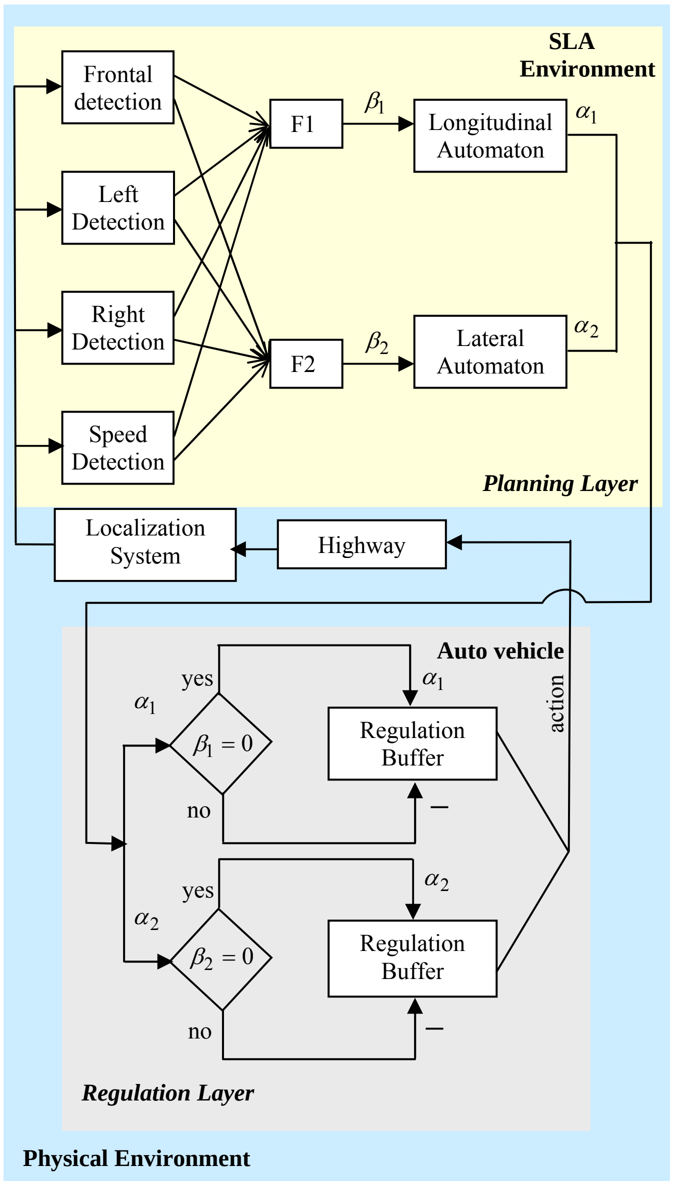

Fig. 2 The model of the Intelligent Vehicle Control System

	Left/Right Sensor Module	
Actions	Vehicle in sensor range or no adjacent lane	No vehicle in sensor range and adjacent lane exists
LINE_OK	0/0	0/0
LEFT	1/0	0/0
RIGHT	0/1	0/0

Table 1 Outputs from the Left/Right Sensor Module

The Headway (Frontal) Module is defined as shown in table 2. If there is a vehicle at a close distance (< admissible distance), a penalty response is sent to the automaton for actions LINE_OK, SPEED_OK and ACC. All other actions (LEFT, RIGHT, DEC) are encouraged, because they may serve to avoid a collision.

	Headway Sensor Module	
Actions	Vehicle in range (at a close frontal distance)	No vehicle in range
LINE_OK	1	0
LEFT	0	0
RIGHT	0	0
SPEED_OK	1	0
ACC	1	0
DEC	0*	0

Table 2 Outputs from the Headway Module

The Speed Module compares the actual speed with the desired speed, and based on the action choosed, send a feedback to the longitudinal automaton.

	Speed Sensor Module		
Actions	Speed: too slow	Acceptable speed	Speed: too fast
SPEED_OK	1	0	1
ACC	0	0	1
DEC	1	0	0

Table 3 Outputs from the Speed Module

The reward response indicated by 0* (from the Headway Sensor Module) is different than the normal reward response, indicated by 0: this reward response has a higher priority and must override a possible penalty from other modules.

7 Implementation of a simulator

In this section is described an implementation of a simulator for the Intelligent Vehicle Control System. The entire system was implemented in Java, and is based on JADE platform.

JADE is a middleware that facilitates the development of multi-agent systems and applications conforming to FIPA standards for intelligent agents. In figure 3 is showed the class diagram of the simulator. Each vehicle has associated an agent, responsible for the intelligent control.

The response of the physical environment is a combination of the outputs of all four sensor modules. The implementation of this combination for each automaton (longitudinal respectively lateral) is showed in figure 4 (the value 0* was substituted by 2).

A snapshot of the running simulator is shown in figure 5.

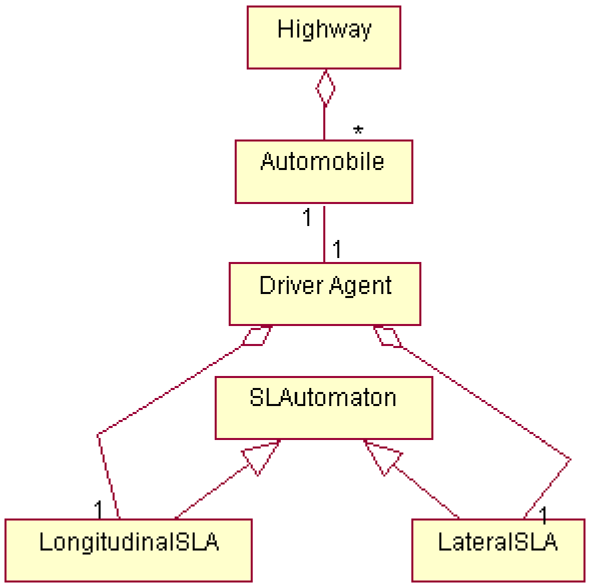

Fig. 3 The class diagram of the simulator

```
// environment response for Longitudinal Automaton
public double reward(int action){
    int combine;
    combine = Math.max(speedModule(action),
                                 frontModule(action));
    if (combine = = 2) combine = 0;
    return combine;
}
// environment response for Lateral Automaton
public double reward(int action){
    int combine;
    combine = Math.max(leftRightModule(action),
                       frontModule(action));
    return combine;
}
```

Fig. 4 The physical environment response

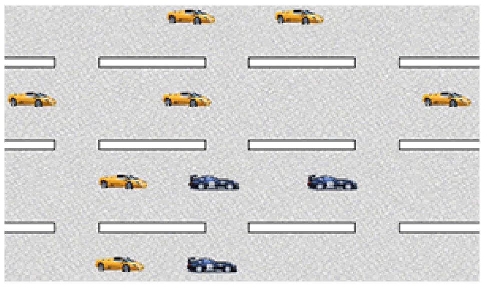

Fig. 5 A scenario executed in the simulator

8 Conclusion

Reinforcement learning has attracted rapidly increasing interest in the machine learning and artificial intelligence communities. Its promise is beguiling - a way of programming agents by reward and punishment without needing to specify how the task (i.e., behavior) is to be achieved. Reinforcement learning allows, at least in principle, to bypass the problems of building an explicit model of the behavior to be synthesized and its counterpart, a meaningful learning base (supervised learning).

The reinforcement scheme presented in this paper satisfies all necessary and sufficient conditions for absolute expediency in a stationary environment. Used within a simulator of an Intelligent Vehicle Control System, this new reinforcement scheme has proved its efficiency.

References:

[1] A. Barto, S. Mahadevan, Recent advances in hierarchical reinforcement learning, *Discrete-Event Systems journal, Special issue on Reinforcement Learning*, 2003.

[2] R. Sutton, A. Barto, *Reinforcement learning: An introduction*, MIT-press, Cambridge, MA, 1998.

[3] O. Buffet, A. Dutech, and F. Charpillet. Incremental reinforcement learning for designing multi-agent systems, In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference onAutonomous Agents*, pp. 31–32,Montreal, Canada, 2001. ACM Press.

[4] J. Moody, Y. Liu, M. Saffell, and K. Youn. Stochastic direct reinforcement: Application to simple games with recurrence, In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium,Technical Report FS-04-02*, 2004.

[5] Cem Ünsal, Pushkin Kachroo, John S. Bay, Simulation Study of Learning Automata Games in Automated Highway Systems, *1st IEEE Conference on Intelligent Transportation Systems (ITSC'97)*, Boston, Massachusetts, Nov. 9-12, 1997

[6] Cem Ünsal, Pushkin Kachroo, John S. Bay, Simulation Study of Multiple Intelligent Vehicle Control using Stochastic Learning Automata, *TRANSACTIONS, the Quarterly Archival Journal of the Society for Computer Simulation International*, volume 14, number 4, December 1997.

[7] Cem Ünsal, Pushkin Kachroo, John S. Bay, Multiple Stochastic Learning Automata for Vehicle Path Control in an Automated Highway System, *IEEE Transactions on Systems, Man, and Cybernetics -part A: systems and humans*, vol. 29, no. 1, january 1999

[8] K. S. Narendra, M. A. L. Thathachar, *Learning Automata: an introduction*, Prentice-Hall, 1989.

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

[10] S. Lakshmivarahan, M.A.L. Thathachar, Absolutely Expedient Learning Algorithms for Stochastic Automata, *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-6, pp. 281-286, 1973