Automatic control based on Wasp Behavioral Model and Stochastic Learning Automata

FLORIN STOICA, DANA SIMIAN
Computer Science Department
“Lucian Blaga” University 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. The aim is to design an automaton, using a reinforcement scheme based on the computational model of wasp behaviour that can determine the best action guided by past actions and environment responses. Using Stochastic Learning Automata techniques, we introduce a decision/control method for intelligent vehicles receiving data from on-board sensors or from the localization system of highway infrastructure.

Key-Words: - Stochastic Learning Automata, Wasp Behavioral Model, 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 response values can be represented in three different models. In the P-model, the response values are either 0 or 1, in the S-model the response values are continuous in the range (0,1) and in the Q-model the values belong to 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.

The algorithm that guarantees the desired learning process is called a reinforcement scheme [5]. 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 [8]. However, these reinforcement schemes are theoretically valid only in stationary environments.

For a nonstationary automata environment resulting from a changing physical environment, we propose a new reinforcement scheme, based on the computational model of wasp behaviour.

The aim of this paper is to define an agent-based model for an Intelligent Vehicle Control System using Stochastic Learning Automata with a learning process driven by a reinforcement scheme with wasp-like behaviour.

The remainder of this paper is organized as follows. In section 2 we present the Stochastic learning automata concept. Sections 3 and 4 contain aspects related to the Intelligent Vehicle Control system. In the section 5 we introduce the Wasp Behavioral Model. Our new reinforcement scheme is presented in section 6. In section 7 is presented an implementation of a simulator for the Intelligent Vehicle Control System and conclusions are presented in section 8.

2 Stochastic learning automata

Mathematically, the environment of a stochastic automaton is defined by a triple \(\{\alpha, c, \beta\}\) where \(\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}\) 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 \( \alpha_i \) 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:
\[
c_i = P(\beta(n) = 1 | \alpha(n) = \alpha_i), \quad i = 1, 2, ..., r
\]

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 [6]. Consider a stationary random environment with penalty probabilities \( \{c_1, c_2, ..., c_r\} \) defined above.

In order to describe the reinforcement schemes, is defined \( p(n) \), a vector of action probabilities:
\[
p_i(n) = P(\alpha(n) = \alpha_i), \quad i = 1, ..., r
\]

We define a quantity \( M(n) \) as the average penalty for a given action probability vector:
\[
M(n) = P(\beta(n) = 1 | p(n)) = \sum_{i=1}^{r} P(\beta(n) = 1 | \alpha(n) = \alpha_i) \cdot P(\alpha(n) = \alpha_i) = \sum_{i=1}^{r} c_i p_i(n)
\]

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:
\[
M(n + 1) < M(n) \quad \text{for all } n [7].
\]

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.

Absolutely expedient learning schemes are presently the only class of schemes for which necessary and sufficient conditions of design are available.

However, the need for learning and adaptation in systems is mainly due to the fact that the environment changes with time. The performance of a learning automaton should be judged in such a context. If a learning automaton with a fixed strategy is used in a nonstationary environment, it may become less expedient, and even non-expedient. The learning scheme must have sufficient flexibility to track the better actions. If the action probabilities of the learning automata are functions of the status of the physical environment (such as in case of an Intelligent Vehicle Control System), the realization of a deterministic mathematical model of this physical environment may be impossible, if not extremely difficult.

For a nonstationary automata environment resulting from a changing physical environment, we propose a new reinforcement scheme, based on the computational model of wasp behaviour.

### 3 Using stochastic learning automata for Intelligent Vehicle Control

In this section, we present a method for intelligent vehicle control, having as theoretical background Stochastic Learning Automata. 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.

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 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 new reinforcement scheme presented in section 6, 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. Our basic model for planning and coordination of lane changing and speed control is shown in figure 1.

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.

<table>
<thead>
<tr>
<th>Left/Right Sensor Module</th>
<th>Actions</th>
<th>Vehicle in sensor range or no adjacent lane</th>
<th>No vehicle in sensor range and adjacent lane exists</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE_OK</td>
<td>0/0</td>
<td>0/0</td>
<td></td>
</tr>
<tr>
<td>LEFT</td>
<td>1/0</td>
<td>0/0</td>
<td></td>
</tr>
<tr>
<td>RIGHT</td>
<td>0/1</td>
<td>0/0</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Headway Sensor Module</th>
<th>Actions</th>
<th>Vehicle in range (at a close frontal distance)</th>
<th>No vehicle in range</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE_OK</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>LEFT</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RIGHT</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SPEED_OK</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ACC</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>0*</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Outputs from the Headway Module

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

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.

<table>
<thead>
<tr>
<th>Speed Sensor Module</th>
<th>Actions</th>
<th>Speed: too slow</th>
<th>Acceptable speed</th>
<th>Speed: too fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEED_OK</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ACC</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 3 Outputs from the Speed Module

4 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.

Fig. 1 The model of the Intelligent Vehicle Control System
5 Wasp Behavioral Model

Theraulaz et al. present a model for self-organization within a colony of wasps ([15]). In a colony of wasps, individual wasp interacts with its local environment in the form of a stimulus-response mechanism, which governs distributed task allocation. An individual wasp has a response threshold for each zone of the nest. Based on a wasp’s threshold for a given zone and the amount of stimulus from brood located in this zone, a wasp may or may not become engaged in the task of foraging for this zone. A lowest response threshold for a given zone amounts to a higher likelihood of engaging in activity given a stimulus. In [16] is discussed a model in which these thresholds remain fixed over time. Later, in [17] is considered that a threshold for a given task decreases during time periods when that task is performed and increases otherwise. In [18], Cicirello and Smith, present a system which incorporates aspects of the wasp model which have been ignored by others authors. They consider three ways in which the response thresholds are updated. The first two ways are analogous to that of real wasp model. The third is included to encourage a wasp associated with an idle machine to take whatever jobs rather than remaining idle. The model of wasp behaviour also describes the nature of wasp-to-wasp interaction that takes place within the nest. When two individuals of the colony encounter each other, they may with some probability interact in a dominance contest. The wasp with the higher social rank will have a higher probability of dominating in the interaction. Wasps within the colony self-organize themselves into a dominance hierarchy. In [18] is incorporated this aspect of the behaviour model, that is when two or more of the wasp-like agents bid for a given job, the winner is chosen through a tournament of dominance contests.

In the following section is presented a new reinforcement scheme for stochastic learning automata.

6 A new reinforcement scheme

In this section we present our approach in implementing a new reinforcement scheme based on a computational model of wasp behavior.

Each stochastic automaton (namely the longitudinal automaton and respectively the lateral automaton) has an associated wasp. Each wasp has a set of response thresholds:

$$\Theta = \{\theta_{w,0}, \ldots, \theta_{w,r}\}$$

where $r$ denotes the number of actions of the automaton (as defined in section 2) and $\theta_{w,j}$ is the response threshold of wasp $w$ for action $i$. An action $i$ broadcast to the automaton a stimulus $S_i$ which is equal to the number of occurrences of action $i$ in the regulation buffer of the automaton. The stochastic automaton will pick up the action $i$ emitting a stimulus $S_i$ with probability:

$$P(i, S_i) = \frac{\theta_{w,j}}{\theta_{w,j} + \theta_{w,i}}$$

The threshold values $\theta_{w,j}$ may vary in the range $[\theta_{\min}, \theta_{\max}]$.

If automaton will execute the action $i$, the threshold $\theta_{w,j}$ is updated as follows:

$$\theta_{w,j} = \theta_{w,j} - \delta_1 \quad (\delta_1 > 0)$$

For each action $j$ other than $i$, the threshold $\theta_{w,j}$ is updated according to:

$$\theta_{w,j} = \theta_{w,j} + \delta_2 \quad (\delta_2 > 0)$$

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 2 (the value $0^*$ was substituted by 2). A snapshot of the running simulator is shown in figure 5.

// 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;
}
Implementation of the learning process for the lateral automaton is described in the following:

```java
// The learning process
public void learning(){
  int i, j;
  double f;
  boolean doIt;
  // choose an action
  i = getAction();
  // compute environment response
  f = reward(i);
  for (int k = 1; k < HISTORY; k++)
    regulation_layer[k-1]=
    regulation_layer[k];
  if (f==0)
    regulation_layer[HISTORY-1] = i;
  else
    // ignore the action!
    regulation_layer[HISTORY-1] = -1;
    doIt=true;
  for (int k = 0; k < HISTORY; k++)
    if (regulation_layer[k]! = i) {
      doIt=false;
      break;
    }
  if (doIt) {
    // all action probabilities
    // become equals
    init();
    switch(i) {
    case LEFT:
      auto.setLane(auto.getLane()-1);
      break;
    case RIGHT:
      auto.setLane(auto.getLane()+1);
      break;
    }
    return;
  }
  // update the action probabilities using a
  // reinforcement scheme based on the
  // wasp-like computational model
  if (f==0){
    if (t[i]-d1 >= t_min) t[i]=t[i]-d1;
    S[i] = getStimulus(i);
    p[i]=S[i]*S[i]/( S[i]*S[i] +t[i]*t[i]);
    for (j=0; j < ACTIONS; j++)
      if (j != i) {
        if (t[j]+d2 <= t_max) t[j]=t[j]+d2;
        S[j] = getStimulus(j);
        p[j]=S[j]*S[j]/( S[j]*S[j] +t[j]*t[j]);
      }
  }
  else{
    if (t[i]+d2 <= t_max) t[i]=t[i]+d2;
    S[i] = getStimulus(i);
    p[i]=S[i]*S[i]/( S[i]*S[i] +t[i]*t[i]);
    for (j=0; j < ACTIONS; j++)
      if (j != i) {
        if (t[j]-d1 >= t_min) t[j]=t[j]-d1;
        S[j] = getStimulus(j);
        p[j]=S[j]*S[j]/( S[j]*S[j] +t[j]*t[j]);
      }
  }
}
```

### 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 new reinforcement scheme presented in this paper is suitable for nonstationary stochastic automata environment, resulting from a changing physical environment. Used within a simulator of an Intelligent Vehicle Control System, this new reinforcement scheme, based on a computational model of wasp behavior, has proved its efficiency.

**Acknowledgment:** This work was completed with the support of the research grant of the Romanian Ministry of Education and Research, CNCSIS 33/2007.

**References:**


