ELSEVIER European Journal of Operational Research 108 (1998) 306–318

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

# Autonomous vehicle navigation using evolutionary reinforcement learning

A. Stafylopatis [*], K. Blekas

*National Technical University of Athens, Department of Electrical and Computer Engineering, 15773 Zographou, Athens, Greece*

Received 15 March 1996; revised 1 August 1997

**Abstract**

Reinforcement learning schemes perform direct on-line search in control space. This makes them appropriate for modifying control rules to obtain improvements in the performance of a system. The effectiveness of a reinforcement learning strategy is studied here through the training of a *learning classifier system* (LCS) that controls the movement of an autonomous vehicle in simulated paths including left and right turns. The LCS comprises a set of condition–action rules (classifiers) that compete to control the system and evolve by means of a genetic algorithm (GA). Evolution and operation of classifiers depend upon an appropriate credit assignment mechanism based on reinforcement learning. Different design options and the role of various parameters have been investigated experimentally. The performance of vehicle movement under the proposed evolutionary approach is superior compared with that of other (neural) approaches based on reinforcement learning that have been applied previously to the same benchmark problem. © 1998 Elsevier Science B.V.

*Keywords:* Learning classifier systems; Genetic algorithms; Reinforcement learning; Autonomous navigation

## 1. Introduction

Typically a trainable adaptive controller architecture facing supervised learning consists of a teacher, the trainable controller and the plant to be controlled. The teacher may be automated as a linear or nonlinear control law, or it may be a human expert. However, in some learning tasks arising in control neither a designed nor a human expert is available. In these situations it may be possible to improve plant performance over time by means of on-line methods performing *reinforcement learning* [1,2], i.e. to adjust a control rule as a function of a measure that in some way evaluates the overall behaviour of the plant. Thus, based on performance-related environmental feedback a mapping is adaptively formed, that associates regions of the environmental state space with different control actions, i.e. the learning system has to discover the most appropriate actions corresponding to the state of the environment as it is perceived through a set of sensors or detectors. Reinforcement learning has been effectively

[*] Corresponding author.

*PII* S0377-2217(97)00372-X

applied to control tasks and, in particular, to the problem of autonomous vehicle navigation in unknown environments [3–6], assuming no prior knowledge about the task and the system to be controlled.

Reinforcement learning addresses the problem of improving performance as evaluated by a measure whose values are supplied to the learning system. In tasks of this kind, based on a scalar evaluation signal, the system has to determine the necessary changes that would lead to increase in the measure of system performance. A reinforcement learning task mainly involves two problems [7]. The first problem is to construct a critic capable of evaluating system performance in a way that is both appropriate to the actual learning objective and informative enough to allow learning over time. The second problem is one of determining how to teach the learning system to provide outputs that improve performance as it is measured by the critic.

The task of providing useful performance information continuously over time is a difficult one and becomes harder when the learning objective is to achieve a terminal state with desired properties. In designing learning systems to solve this problem, the biggest challenge is to account for the link between present actions and future consequences. In many tasks, actions selected earlier in a sequence may turn out to be the most important determinants of eventual outcomes. In such tasks, the information provided to the learning system must not be the *immediate reinforcement* signal associated with the current action, but rather some cumulative measure of reinforcement obtained over many time steps (*delayed reinforcement*). The above problem is referred to as the *temporal credit assignment problem* and concerns the correct assignment of credit or blame to an action when its consequences unfold over time and interact with the consequences of other actions. The general class of *temporal difference* algorithms developed by Sutton [8] aim at solving the problem by training an *adaptive critic* to predict the consequences of an action in the future. Similar techniques are Holland's *Bucket Brigade* algorithm [9] and the *Q-learning* approach introduced by Watkins [10,11].

A direct approach to adjusting control actions in order to improve system performance is to actively explore control space. The need for active exploration stems from the fact that only the evaluation signal is available and not its gradient as a function of the actions. Direct methods for reinforcement learning perform gradient ascent on an evaluation surface whose values are directly available to the learning system. The precise nature of the computation of the evaluation signal by the environment can be anything appropriate for the particular problem and is assumed to be unknown to the learning system. In general, it is some function, deterministic or stochastic, of the input patterns produced by the environment and the output (action) received from the learning system. Since in this approach there is a need for active exploration among possible outcomes, there must be a source of variation allowing exploration of alternative actions.

Our concern in this paper is the use of a direct reinforcement learning strategy to drive an autonomous vehicle through simulated paths made of straight segments as well as left and right turns. The task of providing the proper control commands, so that the vehicle stays on the road and avoids collision, is assigned to a *learning classifier system* (LCS) [9,12–15].

An LCS is a learning system in which a set of condition–action rules called classifiers compete to control the system and gain credit based on reinforcement provided by the environment. Classifiers are generated and modified in an evolutionary process using a genetic algorithm (GA). Survival and evolution depend on the fitness (strength) of individuals, which in turn is based on the cumulative credit of classifiers.

Previous work of ours on the autonomous navigation problem has been based on the use of pure connectionist reinforcement learning schemes assuming no a priori knowledge, as well as on fuzzy-neural approaches [3–5]. In the scheme developed here, a classifier system is trained according to an immediate reinforcement algorithm, while the coverage of the rule-space is achieved by means of an appropriate genetic mechanism.

The autonomous vehicle driving task, which is used here as a benchmark control application, is presented in Section 2. The proposed evolutionary reinforcement learning approach is described in detail in Section 3. Experimental results and performance issues are covered in Section 4. Finally, Section 5 provides conclusions and discussion.

## 2. The autonomous vehicle application

Collision-free autonomous navigation of a vehicle in unknown grounds constitutes an interesting control problem. The objective is to give the proper driving commands as a response to the current state of the vehicle, so that it moves in a course without collisions. This task represents a sensor–motor association problem in the control area. Conventional approaches to solve the problem use artificial intelligence methods (expert systems), which are time-consuming and require much knowledge of the environment in which the vehicle is moving. In our approach very little a priori knowledge is assumed.

The autonomous vehicle uses a number of sensors to perceive its environment. The number and the configuration of the sensors play a very important role in the control task. On the one hand they cannot be too many, and on the other hand their number must be sufficient to provide the necessary information. In our application, eight sensors seem to be adequate to describe the state of the vehicle at each time instant. Fig. 1 shows the location of the sensors on the vehicle, where four of them are placed at the front and two at each side. Each sensor can detect the presence of an obstacle situated within a conic space in front of it and provides a measure of the distance of the obstacle. Although the sensors provide a real-valued distance measure, we have considered for encoding purposes a discrete representation of the distance, by partitioning the cone of each sensor into three logarithmically sized areas, numbered from 0 to 2. A smaller area number corresponds to obstacles at a closer distance. An additional distance value of 3 indicates that there is no obstacle in the range covered by a sensor. Thus, the distinction of four different distance values is possible (very close, close, far, very far), requiring two bits for each sensor value (Fig. 1). Therefore, the environmental state of the vehicle is described by a 16-bit vector.

During navigation, the vehicle is able to perform one of five possible actions in response to the current state. These driving commands are the following:

1. Ahead,

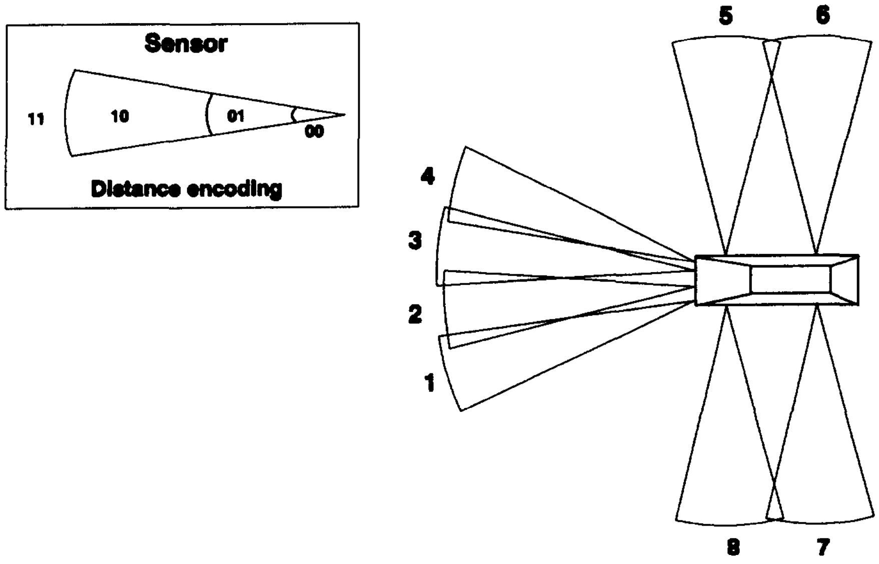

Fig. 1. Position of sensors on the vehicle.

2. 30° right,
3. 60° right,
4. 30° left,
5. 60° left.

The movement of the vehicle is governed by a set of rules that relate states to actions. Each rule $i$ has the form of a pair $(c_i, a_i)$, where the representation of the condition part $c_i$ and the action part $a_i$ is based on the encoding described above. Fig. 2 illustrates a typical training ground.

As already stated, in designing the reinforcement learning system we should effectively deal with the problem related to the performance evaluation of the vehicle in a way that is appropriate to the learning objective, i.e. the movement of the car along a path without collision. Therefore, the evaluation of the vehicle's state should be based on how probable it is for the car to collide and go off the road. This probability can be estimated in terms of the positions of obstacles as they are detected by the sensors. In our application we shall assume that a scalar evaluative signal $r$ (reinforcement) is provided by an external agent (the environment). The learning system is not aware of the mechanism used to compute the signal. The reinforcement is real-valued ranging over [0,1], thus indicating a graded degree of success. One can think of $r = 1$ as "success" and $r = 0$ as "failure". In the control task at hand, failure corresponds to the situation where at least one sensor detects an obstacle at its closest distance. Moreover, success relates to the situation where there is no obstacle in the range covered by any of the front sensors.

The scalar reinforcement signal $r$ is obtained as an average of the partial reinforcements $r_k, (k = 1, \ldots, 4)$, computed by the four front sensors. These partial reinforcements are computed according to the rule $r_k = \xi_k/m$ where the $\xi_k$'s take values from the set $\{0, \ldots, m\}$ and are obtained through a different (linear) discretization of the distances detected by the sensors. (We have used $m = 27$ in our implementation.) The value $\xi_k = 0$ means that sensor $k$ has detected an obstacle right in front of it, while larger values of $\xi_k$ correspond to obstacles at a longer distance. The case of failure is treated in a special manner. If at least one of the eight sensors satisfies the condition $\xi_k = 0$, the global reinforcement assumes a zero value. It is obvious from the above description of the evaluation mechanism that the reinforcement signal takes values in the interval [0,1] and that large reinforcements correspond to actions leading to improved performance of the controlled system.

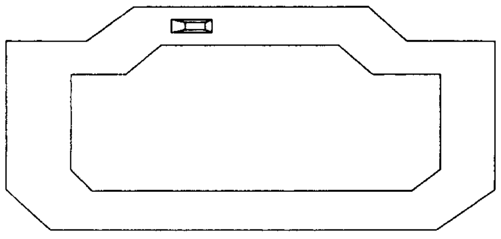
Fig. 2. A typical training ground.

## 3. The learning classifier system

An LCS is a massively parallel, message-passing, rule-based system that is capable of environmental interaction and reinforcement learning through credit assignment and rule discovery [16,17]. Such a production system consists of a set of rules representing a population which evolves through a GA-based learning process. The foundations of LCS were laid by Holland [18]. Typically this class of learning control systems is based on the assumption that each member of the population represents an individual production rule and is referred to as the *Michigan approach* (as opposed to the *Pitt approach* where each member of the population represents a set of rules). Although the LCS presented in the literature exhibits a variety of features, a typical LCS is composed of three main components, as illustrated in Fig. 3.

The *performance system* is the component that interacts with the environment. It comprises input and output interface implemented through detectors and effectors respectively, and a set of rules (classifiers) represented as strings with a condition–action format. A pattern-matching component identifies the classifiers whose condition part is matched by the current detector string

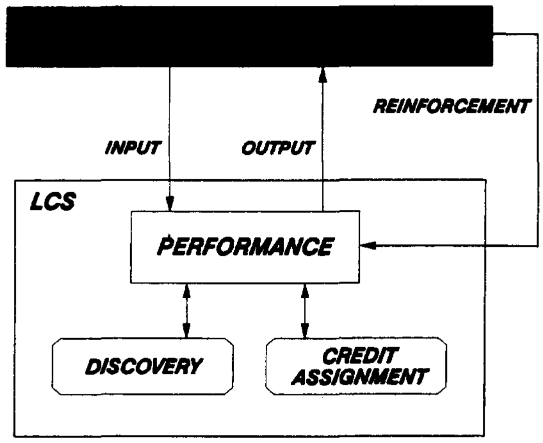

Fig. 3. A diagram of the general LCS approach.

(environmental state). According to some local competition scheme a number of matched classifiers become active and an action is finally chosen that is realized by the effectors. The competition among classifiers can follow complex mechanisms and makes up an inference engine procedure.

The *credit assignment system* is the component of the overall system which performs the activity of learning by means of environmental feedback through adjustment of the conflict-resolution parameters (strengths) of classifiers. Among the variety of credit assignment schemes proposed in the literature, the most widely used are the *Bucket Brigade Algorithm* (BBA) [9], a local incremental learning algorithm, and the *Profit Sharing Plan* (PSP) [19], a global learning scheme with higher performance and higher computational requirements than the BBA.

The third component is the *discovery system*, which develops a learning process by generating new classifiers from the existing classifier set by means of an evolutionary mechanism (GA). The GA is applied with a relatively low frequency and replaces a number of "bad performing" classifiers (typically a small portion of the classifier population) by new ones formed through recombination of high fitness individuals. Traditionally, the strength parameter serves both as a predictor of future payoff related to the credit assignment component and as the fitness of a classifier for the GA, although other measures of fitness have also been considered [15]. Several options, which are related to the general GA optimization approach, such as niching and fitness sharing, can be adopted in the LCS context to accomplish the difficult task of the discovery component [20].

The LCS developed here for the vehicle navigation task represents an adaptation of the general LCS framework, which features exceptions and specificities in several respects. In particular, the proposed scheme, which is described in detail next, incorporates various original characteristics with respect to typical LCS implementations regarding both the credit assignment component and the discovery system.

A schematic representation of the proposed LCS is given in Fig. 4, where the interaction with the environment is performed via sensors for state detection and actuators for motor actions. Each classifier $i$ has a condition part $c_i$ (match string) built upon the binary alphabet plus the "do not care" symbol #, representing a pattern of environmental state, and an action part $a_i$ encoding one of the possible actions. There is a scalar strength $S_i^t$ associated with each classifier $i$ in the current set of working classifiers at time step $t$.

The system starts with a random set of classifiers (population) initialized to some default strength. The condition string of each classifier is initialized by assigning one of the values 0, 1 and # with equal probability to each position, while the action is selected following a uniform distribution. Training of the classifier system consists of a

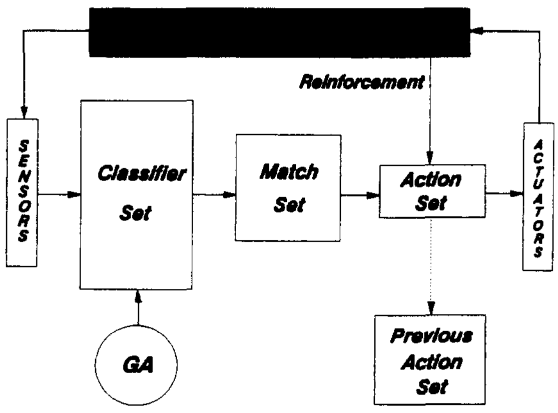

Fig. 4. Classifier system architecture for the navigation task.

sequence of cycles, where each cycle begins with the vehicle at the same initial position and ends with a failure (collision) signal.

At each step during operation, the condition of each classifier in the current set is compared with the detector string. If a classifier's condition matches the detector string in all non-# positions, the classifier is considered close to the environmental state and is included in the current *match set* $M$. Next, an action is selected from among those recommended by the classifiers in $M$. The selection can be done based on the sum of strengths of members of $M$ supporting the action $a_i$, denoted by $S_{M_i}$, either using a roulette wheel method (with sectors sized in proportion to $S_{M_i}$ for each action $a_i$) or using a more flexible Boltzmann probability

$$\mathrm{Prob}(a_i) = e^{S_{M_i}/T} / \sum_j e^{S_{M_j}/T} \tag{1}$$

following some "annealing" parameter $T$. We have adopted the latter scheme in our implementation. The temperature parameter $T$ is decreased by a small fixed percentage each time the GA is applied to the set of classifiers. The appropriate range of values for $T$ depends upon the order of magnitude of the quantities $S_{M_i}$. To bound the behaviour of the exponentials we have restricted the values of the $S_{M_i}$'s in [0,1] by normalizing in $M$. Accordingly, this implies relatively small values for $T$ (initial value less than 10). After attaining some value close to 0 (e.g. the value 0.5), $T$ is kept constant for the rest of the training experiment.

All members of $M$ advocating the selected action become members of the current *action set* $A$. The selected action is performed by the actuators and a reinforcement signal $r$ ($0 \leqslant r \leqslant 1$) is received from the environment.

The operation step is concluded by appropriately doing *credit assignment* to the active classifiers. The algorithm adopted in our approach borrows from ZCS [14], a variant of the BBA, and from techniques used in immediate reinforcement learning schemes [2,3]. Let $S_A^t, S_A^{t+1}$ denote the total strength of the classifiers present in the action set at time steps $t$ and $t + 1$, respectively. Credit assignment is performed by means of the following update equation, where modification of the strength of a set implies suitable apportionment of the increment to all its members:

$$S_A^t \leftarrow (1 - \beta)S_A^t + \eta(r - b)S_A^t + \gamma\beta S_A^{t+1}. \tag{2}$$

The above formulation of the credit assignment algorithm must be interpreted as follows. First a fixed fraction $\beta (0 < \beta < 1)$ of the current strength of each member of the action set is removed. The deducted total amount is divided by the number of classifiers in the current action set, thus yielding an average deduction (per classifier). This amount is then added to the strength of each member of the action set of the previous time step discounted by a factor $\gamma (0 < \gamma < 1)$. This mechanism, which is different from the usual "bucket" paradigm, has proved to be effective in preventing saturation of classifier strengths, thus enhancing the flexibility of the learning system.

On the other hand, the strength of the current set is modified by an amount depending on the received reinforcement $r$, specifically the offset of $r$ with respect to some baseline value $b$, with a learning rate $\eta$. In this way, the reinforcement signal $r$ is rewarding if it is greater than the baseline and is penalizing if it is less than the baseline. The amount $\eta(r - b)S_A^t$ is distributed equally among the classifiers participating in the current action set. The learning parameter $\eta$ generally assumes small values ($\eta \ll 1$) to ensure smooth modification of strengths.

For the baseline either a fixed value (e.g. $b = 0.5$) can be used or a trace $\bar{r}$ of past values of the reinforcement signal, computed as an exponentially weighted average

$$\bar{r}(t) = \lambda\bar{r}(t - 1) + (1 - \lambda)r(t), \tag{3}$$

where $\lambda$ is a decay rate positive and less than 1. This trace can be viewed as a prediction of the expected reinforcement value and can be computed for the classifier system as a whole or for each individual classifier separately. In the latter case, a trace $\bar{r}_i$ is kept for each classifier $i$, according to Eq. (3), and the respective quantity $r - \bar{r}_i$ is used to multiply the portion of $\eta S_A^t$, which is equally distributed among the classifiers participating in the action set. This option of taking individual baselines is intuitively more plausible and has yielded better results in our implementation, since it favours classifiers having better performance over time.

Qualitatively, the above update rule has the effect of rewarding actions which lead to better than usual reinforcement and penalizing actions leading to worse than usual reinforcement. In fact, the use of *reinforcement comparison* leads to faster and more reliable learning. Specifically, an a priori prediction of what reinforcement value to expect on a particular trial is used as the basis for comparison. This prediction computed as an exponentially weighted average of past reinforcement values, is itself adaptive. Convergence in the learning system is achieved because, as learning proceeds and the performance of the system is improved, the reinforcement $r$ received from the environment tends to be equal to the reinforcement trace $\bar{r}$.

In the case of rewarding, the strengths of classifiers in the set difference $M - A$ are decreased by a fixed small fraction $\delta$, to enhance strength differences while preventing unbounded growing of strength. We have taken $\delta$ of the same order as $\beta$.

The *genetic algorithm component* of the classifier system operates in the background and is responsible for generating a number of new classifiers by evolving the existing ones through crossover and mutation, while keeping the total number of classifiers constant. The GA is invoked at cycle ending instants (failure) provided that the interval between two successive invocations exceeds a predefined number of training steps. We have considered both a "panmictic" and a "niching" approach to drive the population evolution. The latter technique is better adapted to the principle of the LCS paradigm. The effect of the GA component is supplemented by a *covering* operation, that is by the insertion of appropriately constructed new rules each time the match set is empty.

The niching approach, that has been adopted in our implementation, is based on creating the same number of new classifiers for each action $a_i$, and using offspring to replace existing classifiers of that action. Classifiers from each niche (subset advocating the same action) are selected for mating using the roulette wheel scheme (with probability proportional to the strength of each classifier). Single-point crossover is applied with a given probability to the condition parts of the mates creating a new individual. Mutation is then applied with a given probability to each position in the offspring (assigning equiprobably one of the other allowed values). Thus, from each pair of mates a new production rule is created having the same action as its parents and strength equal to the average of their strengths. Each newly created classifier replaces a "badly performing" one from the same niche. The latter is again selected by roulette wheel with probability inversely proportional to strength. As reproduction is confined within niches corresponding to actions, the distribution of population members to actions (which is generally taken uniform) is not disturbed by the operation of the GA. Yet actions can be modified by other operations, such as covering and mutation upon failure, that handle special events. These operations offer flexibility and result in a smooth adaptation of the distribution of rules to actions.

*Covering* takes place whenever the match set $M$ is empty. In such a case, a new classifier is constructed whose condition is obtained from the current state vector by randomly setting some components to the "do not care" symbol (with probability less that one-third for each string position). The action part of the new classifier is determined as follows. If the reinforcement received for the last move (which is evaluated as a function of the current state) is relatively large (e.g. greater than 0.7) the "move ahead" driving command is assigned, since it is likely that there are no obstacles in front of the vehicle. Otherwise, one of the remaining actions is randomly chosen. The new classifier then replaces an existing "bad" one selected at random with probability inversely proportional to strength, regardless of action. The strength of the new member is set equal to the average population strength and the reinforcement comparison $\bar{r}_i$ is initialized to zero.

Another case of action modification relates to penalization of the action set whenever a failure occurs. In this case, a mutation operator is applied to all members of the action set $A$, by randomly assigning a new action (different from the current one) to each classifier in $A$. The classifiers keep their strength, which has been normally decreased through application of the credit assignment rule of Eq. (2) with $r = 0$.

## 4. Experimental results

The performance of the proposed system has been investigated through computer simulation experiments. Each experiment consisted of a number of runs that differed only in the seed values for the random number generator. Each run consisted of a sequence of cycles, where each cycle began with the vehicle at the same initial state and ended with a failure signal. At the start of each run the vehicle was placed at a randomly chosen state. The strength of all classifiers is set initially to the same value (100). The condition and action part of each classifier is initially constructed at random (with uniform distribution of actions in the population). The discovery system is invoked at failure instants with a minimum of 100 training steps between successive invocations.

Statistical results of the effectiveness of learning during each run were obtained as follows. For smoothing purposes, at the end of each cycle an average value of the number of steps per cycle was computed by averaging over all cycles from the beginning of the run up to that point. Finally, curves were plotted at the end of each experiment by averaging over all runs. This representation aims at giving an overall view of the progress of learning without being affected by random fluctuations. Of course, under this style of presentation, the contribution of high scores is not readily visualized, since it slowly affects the average value. Several training grounds of varying difficulty have been employed.

The purpose of our experiments is first to study the general behaviour of the proposed evolutionary reinforcement learning mechanism in different grounds and to investigate its dependence upon several critical parameters. On the other hand, we have tried to compare the navigation performance of the vehicle with another reinforcement-based learning approach so as to obtain a better idea on the effectiveness of the proposed scheme.

### *4.1. Parameter considerations*

The learning classifier system described in Section 3 represents an adaptive control model for the collision-free navigation of a vehicle. It involves a number of user-defined parameters which must be appropriately tuned, so as to achieve the best performance. Some of them can be experimentally determined. In general, the range of values of each parameter is specified by constraints depending upon its physical interpretation.

Parameter $\beta$, the fixed fraction deducted from the current strength of each classifier in the action set, must be close to 0. On the other hand, the discount factor $\gamma$ must take a relatively high value but not very close to 1, to prevent unexpected rewards to classifiers which receive small environmental reinforcement. We have found that the best values for the application were $\beta = 0.2$ and $\gamma = 0.8$.

The GA parameters are the probabilities of crossover and mutation used to generate offspring. Following the natural rules of genetic behaviour typical values of these probabilities which have proved to be very efficient are 0.8 and 0.002, respectively. In all the experiments, the annealing parameters $T$ started with an initial value $T = 5.0$ and was reduced by 4% at each application of the GA. After attaining the value 0.5, the parameter $T$ remained unchanged until the end of the experiment. Also, the value of the parameter $\lambda$ used for the computation of the reinforcement comparison $\bar{r}(t)$ (Eq. (3)) was set to $\lambda = 0.9$.

An important issue in classifier systems is the size of the classifier working set (population). As it is difficult to estimate a priori the proper number of rules for a control application, some rules may remain apathetic during the performance stage of system operation. It must be noted here that in our approach this problem is reduced by operations carried out upon collision, where a classifier can change its action, as well as through genetic procedures which can change its condition part. The effect of the choice of population size $N$ is illustrated in Fig. 5 for navigation of the vehicle in the test ground of Fig. 2. The curves represent the average number of steps per cycle as a function of the number of cycles and were obtained for different values of rule population size. The number of the rules cannot be very small ($N = 50$) because this leads to the creation of general-purpose rules which are not able to store sufficient steering information for all possible situations, and, as a

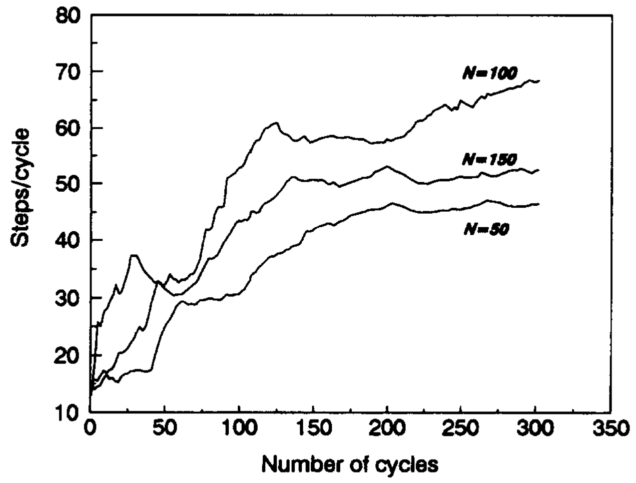

Fig. 5. Performance of LCS for different population sizes.

consequence, the rule discovery mechanism falls into a looping behaviour. On the other hand, a very large value of population size ($N = 150$) has not proved to be efficient, since this directs the system towards a slow learning behaviour. An appropriate value found for the number of classifiers was $N = 100$.

The learning rate $\eta$ plays an important role in the LCS approach. As can be observed from Eq. (2), this parameter specifies the amount of reinforcement added to the strength of the current action set. This amount must not be excessively high or low because this will lead to an unstable behaviour during learning. The modifications of strengths must follow a smooth and steady update process. Fig. 6 shows the effect of the rate $\eta$ to the navigation of the vehicle in the ground of Fig. 2. The three curves represent the average number of steps per cycle as a function of the number of cycles for three values of $\eta$. The driving performance in the case $\eta = 0.1$ is too slow, while it is unstable in the case $\eta = 0.2$. The best behaviour was obtained for $\eta = 0.15$.

### 4.2. Comparative results

The above parameter values have proved to be very effective for the performance of the learning classifier system during collision-free movement of the vehicle through different test grounds. In order to evaluate the overall performance of our model in the given control task, we have compared it with a connectionist reinforcement learning approach [3]. The latter is a modification of a general class of reinforcement learning algorithms, called REINFORCE algorithms, that have been developed by Williams [2]. It is used to train the weights of a neural network consisting of Bernoulli logistic units with no hidden layers, whose outputs take values in $\{0,1\}$. The weight update rule is given by the following equation assuming that $w_{ij}$ indicates the weight between input $x_j$ and output $y_i$ of the connectionist network

$$\Delta w_{ij} = a(r - b)\bar{e}_{ij}, \tag{4}$$

where $a$ is a learning rate factor, $r$ is the reinforcement signal delivered by the environment and $b$ is a reinforcement baseline. The factor $\bar{e}_{ij}$ denotes a trace of the *characteristic eligibility* $e_{ij}$ of $w_{ij}$. The latter is defined as $e_{ij} = \partial \ln g_i / \partial w_{ij}$, where $g_i$ is the probability distribution function of the output $y_i$ of the $i$th unit.

In the case of Bernoulli logistic units, the probability $p_i$ that unit $i$ will produce an output equal to 1 (parameter of the Bernoulli distribution) is a logistic function of the weighted average $\sum_j w_{ij}x_j$

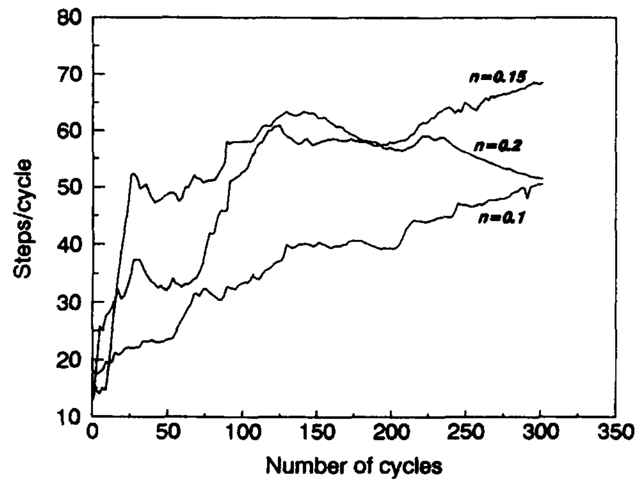

Fig. 6. Performance of LCS for different values of $\eta$.

that constitutes the input to unit $i$. Computation of the characteristic eligibility yields

$$e_{ij} = (y_i - p_i)x_j. \tag{5}$$

The trace $\bar{e}_{ij}$ can be computed as an exponentially weighted average of recent values of $e_{ij}$ by means of the difference equation

$$\bar{e}_{ij}(t) = \mu\bar{e}_{ij}(t-1) + (1-\mu)e_{ij}(t), \tag{6}$$

where $\mu$ is a decay factor positive and less than 1.

The above learning rule, which constitutes a modification of the REINFORCE model as originally defined, produces a kind of temporal credit assignment in the network behaviour. As a consequence, it succeeds in changing the weights of the network in a smoother manner than in the original REINFORCE algorithms, thus improving performance.

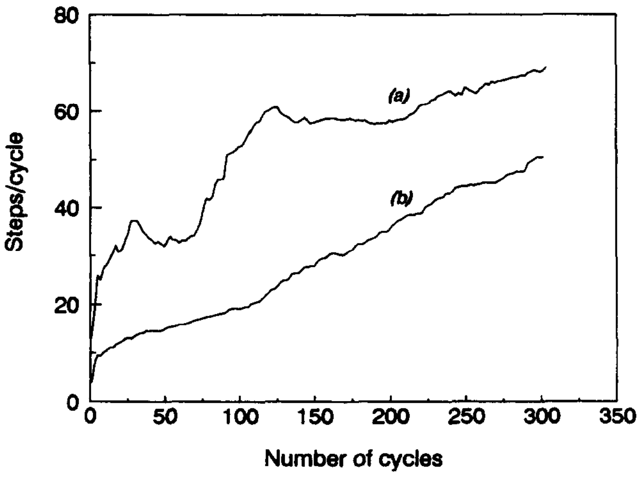

Fig. 7. Performance comparison of LCS (a) and eligibility trace algorithm (b).

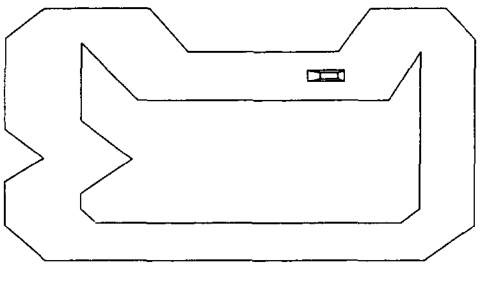

Fig. 8. Performance of LCS in a difficult ground.

Fig. 7 illustrates the behaviour of vehicle navigation under the learning classifier system (a), and under the eligibility trace version of the REINFORCE algorithm (b), in terms of the average value of the number of steps per cycle. The two curves were obtained from experiments on the ground depicted in Fig. 2. In the case of the eligibility trace scheme, the values of the parameters were $b = 0.5$, $a = 0.9$ and $\mu = 0.9$. The parameters for the learning classifier system assumed the best values found in the experiments described previously. As can be observed from this figure, clearly the proposed evolutionary approach performs better. The learning classifier model is more powerful and has the ability to construct appropriate control rules with no a priori knowledge of the dynamics associated with the steering behaviour of the vehicle. Moreover, through its genetic aspect, it has the advantage to explore effectively the rule space and avoid the problems of convergence related to the pure reinforcement learning schemes.

Fig. 8 displays the performance of the proposed learning classifier system in the ground shown in Fig. 9, which can be considered a "difficult" one. It is obvious from this curve that the model is able to create effective driving rules for navigation of the vehicle in this kind of "hard-to-learn" ground. It must also be noted that in such test grounds the eligibility trace algorithm has bad performance and is not able to attain a steady behaviour.

## 5. Conclusions

We have addressed the problem of autonomous vehicle navigation using a LCS approach. Starting from the formulation of the problem our study focused on the development and exploration of the appropriate learning model for control. The proposed evolutionary learning scheme borrows both from genetic learning classifier systems and reinforcement learning and aims to construct efficient rules associated with the steering behaviour of the vehicle. In addition, it has the capability of using the reinforcement learning aspect in a

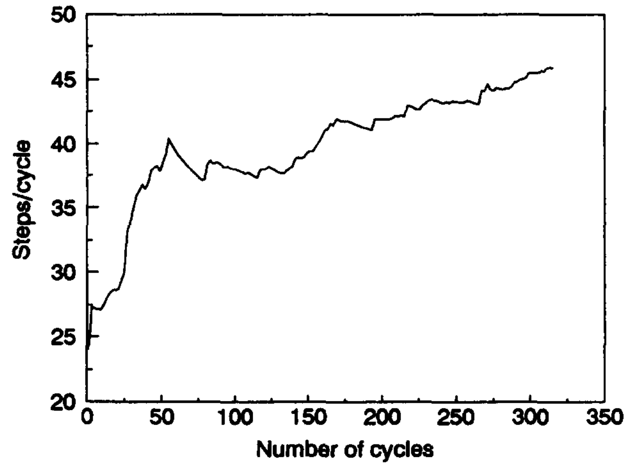

Fig. 9. A "difficult" training ground.

more efficient way than other learning classifier approaches, obtaining characteristics that enhance its learning performance.

We have experimentally evaluated the proposed system on several training grounds of varying difficulty and obtained statistical results of the effectiveness of learning during each experiment, by computing a moving average of the number of steps per cycle. These experiments allowed us to investigate the role of several parameters that affect the behaviour of the system, and determine their most appropriate values for the application. To illustrate the effectiveness of our evolutionary approach we have compared it with a neural network scheme based on reinforcement learning that had been applied to the same benchmark application in previous work of ours. The results have shown that the evolutionary reinforcement learning mechanism exhibits clearly superior performance and offers a particularly promising alternative in adaptive movement control.

Several extensions and modifications to the presented approach may be considered in the future in order to further investigate the use of learning classifier systems in this type of application. On the one hand, we can explore the reinforcement learning aspect by selecting among several known techniques, in order to achieve better credit assignment mechanisms. On the other hand, we can improve the behaviour of the discovery system by considering hybrid techniques that can be effectively applied to overcome the weaknesses of GA.

An interesting alternative is to consider the use of a *fuzzy learning classifier system* (FLCS), i.e. a type of generic-based machine learning system whose classifier list is a fuzzy rule bank and which creates fuzzy rules working in a fuzzy environment.

Another direction of future investigation is to incorporate an *adaptive critic element* (ACE) into the system. This element can be a neural network, that learns to produce a prediction of future reinforcement for each state. Thus, the network may evaluate the action suggested by the classifier system by computing an internal reinforcement signal, which is taken into account upon updating the strengths of classifiers in the action set. Referring to the general schema of Fig. 3, which presents an abstract view of the learning classifier system, we can add a box describing the effect of the ACE which can communicate both with the environment and the learning classifier box.

## References

[1] R.S. Sutton, A.G. Barto, R.J. Williams, Reinforcement learning in direct adaptive optimal control, IEEE Control Systems Mag. 12 (2) (1992) 19–22.

[2] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning 8 (1992) 229–256.

[3] D. Kontoravdis, A. Stafylopatis, Reinforcement learning techniques for autonomous vehicle control, Neural Network World 3 and 4 (1992) 329–346.

[4] D. Kontoravdis, A. Likas, A. Stafylopatis, Collision-free movement of an autonomous vehicle using reinforcement learning, in: Proc. 10th Europ. Conf. on Artificial Intelligence, Wiley, Vienna, Austria, 1992, pp. 666–670.

[5] D. Kontoravdis, A. Likas, K. Blekas, A. Stafylopatis, A Fuzzy Neural Network Approach to Autonomous Vehicle Navigation, in: Proc. Europ. Robotics and Intelligent Systems Conf., Malaga, Spain, 1994, pp. 243–252.

[6] J. del R. Millan, C. Torras, A reinforcement connectionist approach to robot path finding in non-maze-like environments, Machine Learning 8 (1992) 363–395.

[7] W.T. Miller, R.S. Sutton, P.J. Werbos (Eds.), Neural Networks for Control, MIT Press, Cambridge, 1990.

[8] R.S. Sutton, Learning to predict by the methods of temporal differences, Machine Learning 3 (1988) 9–44.

[9] D.E. Goldberg, Genetic algorithms in search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[10] M. Dorigo, H. Bersini, A comparison of Q-learning and classifier systems, in: From Animals to Animats 3: Proc. of the Third Int. Conf. on Simulation of Adaptive Behavior, MIT Press, Cambridge, MA, 1994, pp. 248–255.

[11] C.J.C.H. Watkins, Learning from Delayed Rewards, Ph.D. Thesis, Cambridge University, England, 1989.

[12] R. Abu Zitar, M.H. Hassoun, Neurocontrollers trained with rules extracted by a genetic assisted reinforcement learning system, IEEE Trans. on Neural Networks 6 (4) (1995) 859–879.

[13] S.W. Wilson, A classifier system and the animat problem, Machine Learning 2 (1987) 199–228.

[14] S.W. Wilson, ZCS: A zeroth level classifier system, Evolutionary Computation 2 (1) (1994) 1–18.

[15] S.W. Wilson, Classifier fitness based on accuracy, Evolutionary Computation 3 (2) (1995) 149–175.

[16] L.B. Booker, D.E. Goldberg, J.H. Holland, Classifier systems and genetic algorithms, Artificial Intelligence 40 (1989) 235–282.

[17] O. Cordón, F. Herrera, E. Herrera-Viedma, M. Lozano, Genetic Algorithms and Fuzzy Logic in Control Processes, Technical Report #DECSAI-95109, Universidad de Granada, Spain, 1995.
[18] J.H. Holland, K.J. Holyoak, R.E. Nisbett, P.R. Thagard, Induction: Processes of Inference, Learning, and Discovery, MIT Press, Cambridge, MA, 1986.
[19] J.J. Grefenstette, Credit assignment in rule discovery systems based on genetic algorithms, Machine Learning 3 (1988) 225–245.
[20] J. Horn, D.E. Goldberg, K. Deb, Implicit niching in a learning classifier system: Nature's way, Evolutionary Computation 2 (1) (1994) 37–66.