Genetics-Based Machine Learning and Behavior-Based Robotics: A New Synthesis

Marco Dorigo, *Member, IEEE*, and Uwe Schnepf

***Abstract*— Intelligent robots should be able to use sensor information to learn how to behave in a changing environment. As environmental complexity grows, the learning task becomes more and more difficult. This problem is faced using an architecture based on learning classifier systems and on the structural properties of animal behavioral organization, as proposed by ethologists. After a description of the learning technique used and of the organizational structure proposed, experiments that show how behavior acquisition can be achieved were presented. The simulated robot learns to follow a light and to avoid hot dangerous objects. While these two simple behavioral patterns are independently learned, coordination is attained by means of a *learning coordination* mechanism. Again this capacity is demonstrated by performing a number of experiments.**

I. Introduction

The traditional knowledge-based approach to artificial intelligence shows some fundamental deficiencies in the generation of powerful and flexible reasoning techniques. Explaining the cognitive abilities of the brain purely in terms of symbol manipulation as in current AI implementations seems to lack the flexibility and expressiveness of natural cognitive systems. Behavior-based robotics claims to provide a better—and, perhaps, the only possible—way to develop intelligent systems [5].

Most of the work done in behavior-based robotics focuses on the design of appropriate robot behavior, hoping that powerful coordination techniques (e.g., subsumption architecture [4], action selection model [6], [16]) lead to more complex behavioral sequences, in this way providing flexibility and robustness to the robot's overall behavior.

We believe that these approaches are insufficient as long as the adaptation takes place only in the mind of the designer of such an autonomous system. An autonomous agent must possess this adaptive power itself in order to adapt its behavior to any changes in the environment. Nature has produced such adaptability by means of evolution. Natural systems have genetically learned to adapt, i.e., to increase the likelihood to survive and to have more offspring. This evolutionary process has finally led to neural learning, a flexible way of adaptation, and to cognitive abilities. Only if we can reproduce these adaptation processes, we will be able to understand the emergence of cognitive skills. For this reason, we consider genetics-based learning as a plausible and powerful way to develop intelligent systems.

We have developed an approach that is based on both ethological and evolutionary considerations. In this way, we intend to construct a model of cognition as a biological phenomenon that serves only one goal: to increase the chances of a species to survive. The approach we present in this paper is considered to reflect these basic mechanisms and to produce the kind of adaptability necessary for robust and flexible robot intelligence. The paper is organized as follows. Section II deals with the principles of genetics-based machine learning and Section III with behavior-based robotics. In subsequent sections we present our research methodology and provide implementational details on the system developed, together with results. Section IV describes the architecture of the system. In Section V our approach is compared to related work. Section VI introduces the experiments and the results obtained, together with discussion and finally, in Section VII, we sketch future work to be done.

Manuscript received February 24, 1991; revised May 26, 1992.

M. Dorigo was with the Politecnico di Milano Artificial Intelligence and Robotics Project, Dipartimento di Elettronica e Informazione, Via Ponzio 34/5, 20133 Milano, Italy, and is now with the International Computer Science Institute, 1947 Center Street, Ste. 600, Berkeley, CA 94704-1105.

U. Schnepf is with the AI Research Division, German National Research Center for Computer Science (GMD), P.O. Box 1316, 5205 Sankt Augustin 1, Germany.

IEEE Log Number 9205785.

II. Genetic Algorithms, Learning Classifier Systems and Genetics-Based Machine Learning

Genetics-based machine learning (GBML) systems are a class of learning systems that learn to accomplish a task by means of interaction with an environment. They interact with the environment, monitoring it by means of sensors and acting according to received messages. The learning process is guided by feedback about the quality of actions. Therefore, they belong to the class of *reinforcement learning* systems (see Fig. 1). The name genetics-based machine learning stems from the algorithm used to implement rule discovery (the genetic algorithm). In our work we used a particular kind of GBML system known as learning classifier systems (LCS). It presents the following peculiarities.

- Rules are strings of symbols over a three-valued alphabet ($A = \{0, 1, ^*\}$) with a condition→action format (in our system each rule has two conditions that have to be simultaneously satisfied in order to activate the rule).
- A limited number of rules fire in parallel.
- A pattern-matching and conflict-resolution subsystem identifies which rules are active in each cycle and which of them will actually fire.

In LCSs we can observe two different learning processes. In the first one, the set of rules is given and its use is

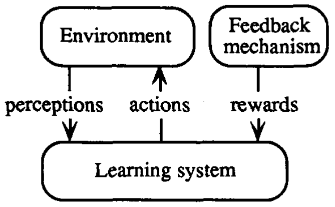


Fig. 1. A general reinforcement learning model.

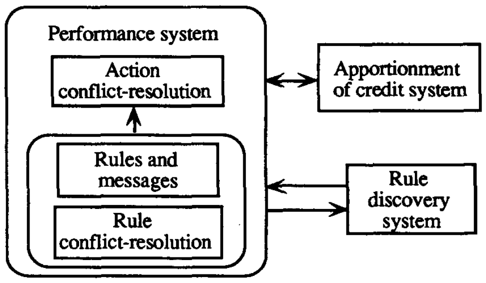


Fig. 2. Structure of a GBML system.

learned. In the second, new and possibly useful rules can be created. These two kinds of learning are accomplished by the apportionment of credit algorithm and by the rule discovery algorithm respectively. New rules are learned using past experience and the way to use them is learned using environmental feedback.

The system is then composed of the following three main parts (as illustrated in Fig. 2).

- The performance system.
- The apportionment of credit system.
- The rule discovery system.

In the following, we briefly introduce the three subsystems.

A. The Performance System

The performance system is composed of (see Fig. 3) the following.

- A set of rules, called classifiers.
- A message list, used to collect messages sent from classifiers and from the environment to other classifiers.
- An input and an output interface with the environment (detectors and effectors) to receive/send messages from/to the environment.
- A feedback mechanism to reward the system when a useful action is performed and to punish it when a wrong action is done.

Initially, at time t_o, a set of classifiers is created (they may be generated randomly or by some algorithm that takes into account the structure of the problem domain) and the message list is empty. At time t_1 environmental messages are appended to the message list, which are matched against the condition part of classifiers, and these matching classifiers are set to active status. At time t_2 messages coming from the environment and messages sent by classifiers active at time t_1 are appended to the message list. They are then matched

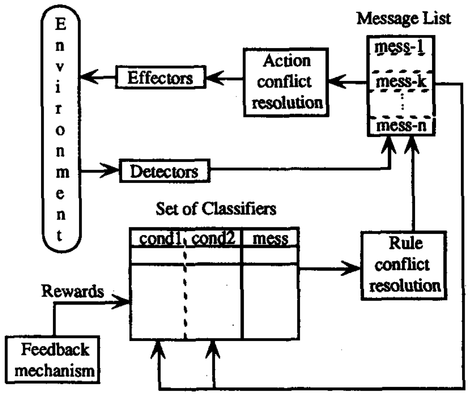


Fig. 3. The performance system.

against classifiers in the classifier set, and matching classifiers become activated. The message list is then emptied and the cycle repeated, from t_1.

The need for a rule conflict-resolution system is one of the reasons for the introduction of an apportionment of credit algorithm that redistributes environment payoff to the rules that caused the performed actions. This allows the performance system to choose which rules to fire in accordance to some measure of their usefulness. Conflict resolution must also be used to solve conflict when effectors propose inconsistent actions (e.g., "go right" and "go left").

In order to explain better the nuances of this performance system, let us introduce some terminology.

- A *classifier* (rule) is a string composed of three *chromosomes,* two chromosomes being the condition part,[1] the third one being the message/action part; we will call a classifier an *external classifier* if it sends messages to the effectors, an *internal classifier* if it sends messages to other classifiers.
- A *chromosome* is a string of n positions; every position is called a *gene.*
- A *gene* can assume a value, called *allelic value,* belonging to an alphabet that is usually $A = \{0, 1, *\}$. (The reasons underlying this choice are to be found in the rule discovery algorithm used, namely the Genetic Algorithm. In fact, it has been demonstrated [13] that the lower the cardinality of the alphabet, the higher the efficiency of the algorithm in processing useful information contained in the structure of the chromosomes. This is discussed in the next section.)

Consider for example the following classifier:

$$*1*;011 \rightarrow 010.$$

It consists of the two conditions *1* and 011, and the action 010. The second condition is matched only by the message 0 1 1, while the first one is matched by any message with a 1 in the second position. The * symbol stays for "don't care,"

[1]Although we use in our example only two chromosomes, it is in general possible to utilize any number n of chromosomes in the conditions part of a classifier ($n \geq 2$), without changing the representational power of the resulting system.

that means both symbols, 0 or 1, match the position. If both the conditions are matched by some message, then the rule is activated and the message/action part, i.e. the chromosome 0 1 0 in the example, is appended to the message list at the subsequent step; some of the messages on the message list can be external messages: after conflicts are solved in the action conflict resolution box, they are sent to the effectors.

B. The Genetic Algorithm

Genetic algorithms are a class of stochastic algorithms that has been successfully utilized both as an optimization device and as a rule-discovery mechanism. They work modifying a population of solutions (in GBML a solution is a classifier) to a given problem. Solutions are properly coded and a function, usually called the *fitness function*, is defined to relate solutions to performance. The value returned by this function is a measure of the solution quality. The fitness of a classifier is determined by its usefulness calculated with an apportionment of credit algorithm instead of a fitness function.

GAs work as follows:

Let P be a population of N chromosomes (*individuals* of P). Let $P(0)$ be the initial population, randomly generated, and $P(t)$ the population at time t. The main loop of a GA consists in generating a new population $P(t + 1)$ using the existing one $P(t)$, applying some so-called *genetic operators*. These operators modify randomly chosen individuals of population $P(t)$ into new ones. The two most important of these operators are *crossover*—it recombines individuals by taking two of them, cutting in at a randomly chosen positions and recombining the two in such a way that some of the genetic material of the first individual goes to the second one and vice versa—and *mutation* that randomly changes some of the values of the genes constituting an individual. This way, the population $P(t + 1)$ is created by means of a *reproduction* operator that gives higher reproduction probability to higher fitted individuals. The overall effect of GAs' work is to move the population P towards areas of the solution space with higher values of the fitness function.

The computational speed-up that we obtain using GAs with respect to random search is due to the fact that the search is directed by the fitness function. This direction is not based on whole chromosomes, but on their parts that are strongly related to high values of the fitness function: these parts are called *building blocks* [10], [13]. It has been proven [22] that GAs process at each cycle a number of building blocks proportional to the number of individuals of the population. GAs are therefore useful for every problem where an optimal solution may be obtained as composition of a collection of building blocks.

To use GAs as a rule-discovery system means to hypothesize that new and more useful rules can be created by recombination of other less useful ones. In order to preserve the system performance the GA is allowed to replace only a subset of the classifiers. The worst m classifiers are replaced by m new classifiers created by the application of the GA on the population. The new rules are tested by the combined action of the performance and apportionment of credit algorithms. Since testing a rule requires many time steps, GAs are applied with a lower frequency unlike the performance and apportionment of credit systems.

C. The Apportionment of Credit System

The main task of the apportionment of credit algorithm is to classify rules in accordance with their usefulness. In other words, the algorithm works as follows: a time varying real value called *strength* is associated to every classifier C. At time zero each classifier has the same strength. When an external classifier causes an action on the environment a payoff is generated whose value is dependent on how good the action performed was with respect to the system goal. This reward is then transmitted backward to internal classifiers that caused the external classifier to fire. The backward transmission mechanism, examined in detail later, causes the strength of the classifiers to change in time and to reflect their relevance to the system performance (with respect to the system goal).

It is not possible to keep track of all the paths of activation actually followed by the rule chains (a rule chain is a set of rules activated in sequence, starting with a rule activated by environmental messages and ending with a rule performing an action on the environment) because the number of these paths grows exponentially. It is then necessary to have an appropriate algorithm that solves the problem using only local (in time and space) information.

Local in time means that the information used at every computational step is coming only from a fixed recent temporal interval. Spatial locality means that changes in a classifier strength are caused only by classifiers directly linked to it; classifiers C_1 and C_2 are linked if the message posted by C_1 matches a condition of C_2.

The classical algorithm used for this purpose is the *Bucket Brigade* algorithm [3]. This algorithm models the classifier system as an economic society, in which every classifier pays an amount of its strength to get the privilege of appending a message to the message list and receives a payment by all classifiers activated because of the presence in the message list of the message it appended during the preceding time step. In this way payoff flows backward from the environment again to the environment through a chain of classifiers. The net result is that classifiers that participate in chains that cause high rewarded actions tend to increase their strength. A good introduction to GAs and LCSs, explaining the basic algorithms in more detail, can be found in [10].

III. Behavior-Based Robotics

The main idea in the approach of behavior-based robotics as an alternative to the traditional knowledge-based AI is that intelligent behavior cannot be created in artificial systems without the ability to interact with a dynamically changing unstructured environment. Cognition emerges only when autonomous systems try to impose structure on the perceived environment in order to survive. These structures in turn provide the substratum for more intelligent behavior: the skills to learn, to generalize and to abstract from given information, to form categories and concepts, the emergence of goal-directed behavior, the creation of internal world models, and the development of problem-solving techniques. These basic cognitive skills are unlikely to have been present in biological autonomous systems from the very beginning of life. They are more likely to have developed as part of the evolutionary process. We are interested in reconstructing this process in order to create robot intelligence.

1) Designing Behaviors: Most of the work done in the field of behavior-based robotics so far focuses on the design of appropriate behavioral modules and coordination techniques to combine these behaviors in the most fruitful way in order to enable the autonomous system to perform flexible and robust actions. As most roboticists are interested in a particular robot task (e.g. avoiding obstacles, following walls, passing doors, grasping objects), the behaviors designed to fulfill these tasks are well tailored to a particular situation. So far, no conceptual model of how behaviors are related to each other has been presented. Although some recent work has been reported on the use of genetic algorithm techniques within this approach [7], we do not expect this more engineering-oriented approach to behavior-based robotics to give any deeper insights in the evolutionary processes mentioned above. We have to look for other methodologies to explain the emergence of adaptive behavior.

2) Ethological Models: During the last 80 years, researchers working in Ethology have tried to answer the very same questions that we consider of great importance for the explanation of goal-directed behavior. The explanation models of behavioral organization in animals or human beings presented so far in ethology cannot cover the many different aspects of adaptive behavior and have not been tested intensively in the context of autonomous systems. However, we believe that these models are better suited as explanation models than the engineered ones based on the robot task, as the ethological models are based on the intensive observation of animal and human behavior.

3) Our Approach: The Tinbergen model [18] is a model of animal behavior that we consider of great utility for our approach. In general one can describe this model as a hierarchy of behavioral modules or so-called *instinct centers.* Each instinct center is decomposed into more fine-grained behavioral sequences represented by instinct centers at the next lower level.

The instinct centers at the same level of the hierarchy compete against each other in becoming active. At a given level of the hierarchy only the instinct center having high excitational status can activate the fine-grained behavioral sequences at the level below it. The excitational status of an instinct center is influenced by the excitation coming from inner and outer sensors, from motivations and from instinct centers noted previously. So-called *innate releasing mechanisms* directly related to each individual instinct center prevent the instinct center from arbitrarily entering the competition for control over the agent.

Only if a particular threshold value has been achieved, the excitation is released. This mechanism serves to eliminate chaotic behavior in the behavioral organization. A more detailed description of the model can be found in [18]. Fig. 4 illustrates the hierarchical relation between the various instinct centers.

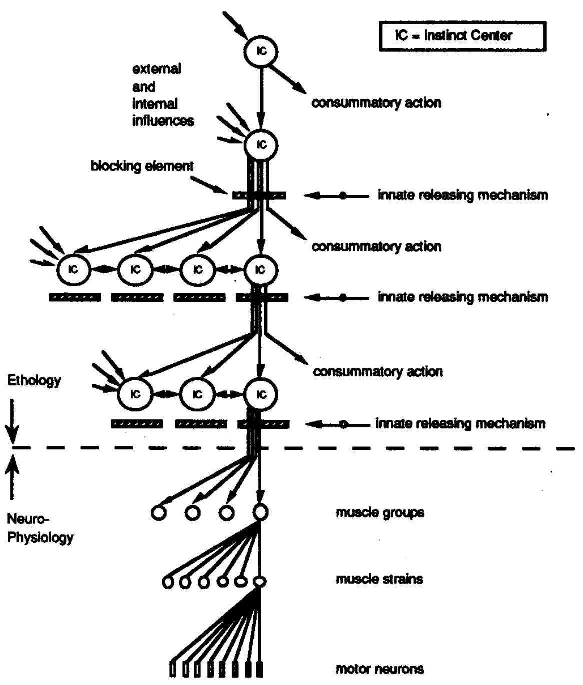


Fig. 4. A hierarchy of instinct centers.

In the following section we will describe our implementation of the Tinbergen model.

IV. The System Architecture

The Tinbergen model described in Section III served as the starting point for our implementational model. We have developed a system architecture based on the model features that has been implemented on a transputer system. The following paragraphs will briefly sketch what we call the *complete model,* which represents to a large extent the functionalities of the Tinbergen model. The current implementation will be described by the so-called *current model,* which represents a subset of the complete model. Over the course of time, the current model is expected to progressively approach the complete one.

A. The Complete Model

Our system consists of many classifier systems running in parallel. Each classifier system learns a simple behavior through interaction with the environment, the system as a whole has as its learning goal the coordination of activities. The hierarchical organization allows us to distinguish between two different learning activities: the learning of behavioral sequences and the learning of coordination sequences. The classifier systems at the lowest level of our hierarchical model learn behavioral sequences, i.e., real actions activated by sensory input from the environment, whereas the classifier systems at higher levels learn to coordinate the activities of

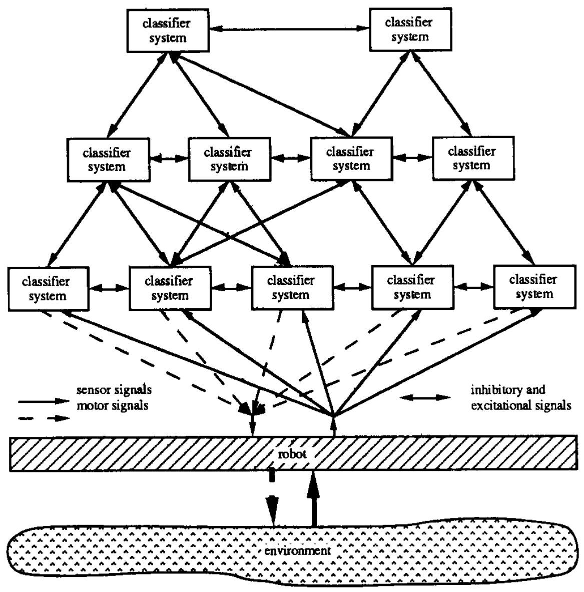


Fig. 5. The complete model.

classifier systems at the level below. Hence, only classifier systems at the lowest level have direct access to the environment via the robot's sensors and actuators. Fig. 5 illustrates the hierarchical organization of the model.[2] Note, that it does not represent a hierarchical system in the classical computational sense, since there is no hierarchical flow of control and no explicit input/output relation between different modules involved in this model. Only the increasing complexity in behavioral organization is hierarchical.

The complete model can be characterized in the following way.

1) Classifier systems are working in parallel at any level of the hierarchy.
2) Each classifier system at the lowest level represents one possible class of interactions with the environment.
3) Each classifier system at any higher level represents one possible class of interactions among classifier systems at the level below.
4) Classifier systems at the same level can be associated with one common classifier system at the higher level when they are simultaneously active in a given situation.
5) Each classifier system receives excitational and inhibitory signals from connected classifier systems and computes its activational status. If the activational status is high enough to activate the innate releasing mechanism, the classifier system becomes active and sends appropriate messages.
6) Classifier systems at the same level of the hierarchy compete against each other in becoming active by exchanging inhibitory signals according to their excitational status.
7) Only classifier systems at the lowest level have access to the sensors and actuators of the robot.
8) The motor signals of all active classifier systems at the lowest level are collected to calculate a weighted sum of motor signals.
9) Excitational signals of a classifier system are sensor signals (only at the lowest level), motivational stimuli, and excitational signals from associated classifier systems in the level above, whereas inhibitory signals come from associated-(neighboring)-classifier systems at the same level.
10) Classifier systems at higher levels receive activational information from and send excitational or inhibitory signals to the associated classifier systems at the next level below.
11) Classifier systems can be associated with more than one classifier system at the next higher level.

[2]By calling it the "complete" model, we refer to the complete range of functionality that reflects the functionality of the underlying Tinbergen model. However, we do not refer to the completeness of the behavioral complexity that is unspecified in general.

What we describe is a construction process. The associative processes that correlate classifier systems commonly active in a given situation were not contained in the Tinbergen model, and it does not have a winner-take-all strategy. The motor signals emitted by each low-level classifier system are weighted signals and summed up to calculate the actual motor signals. This approach is quite similar to the approach developed by Arkin for his schema-based mobile robot navigation [1], or the one of Holland and Snaith, who are using such a technique in their manifold architecture [15].

Another important feature is the extendibility of the system: each time it encounters a novel situation for which no appropriate learned behavior exists, the system instantiates a new classifier system that should learn to deal with this new situation.

B. The Current Model

What we have described so far is the complete model. In a simulation of the current model, the system characteristics 1, 2, 3, 7, 8, 10 have been implemented. See Fig. 6 for an illustration of the current model. Further system characteristics are as follows.

- The classifier systems at the lowest level have been associated by design with the classifier system at the next level.
- The classifier systems at the lowest level receive specific and different sensor signals (no other signals) and compute their motor signals. They do not compete against each other in becoming active.

V. Related Work

Our work finds its cultural background in three major research areas. We have integrated ideas developed in the disciplines of ethology, machine learning and behavior-based robotics to build a framework we believe to be general enough to be used for developing learning robots. Ethology and robotics are represented in our approach by the work of Tinbergen and of Brooks and his group at MIT. Their

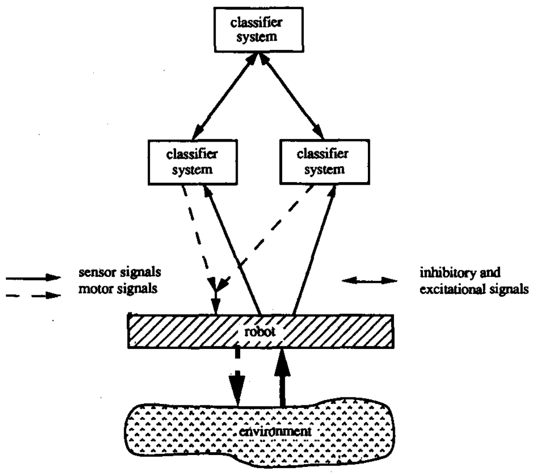


Fig. 6. The current model.

importance has already been sufficiently stressed in preceding sections and elsewhere [17]. The ideas developed in the field of machine learning firstly by Holland [13], [14], and then formalized under the term "*Animat problem*" by Wilson [19] have been useful to our approach. The Animat problem is the problem faced by an artificial animal that has to learn how to survive in its environment. Wilson proposes the following reasons to explain why this is a difficult problem: information is difficult to classify because there is no a priori knowledge about how to relate environmental situations, i.e., information coming from the environment, to actions that can take the Animat closer to the goal state and also because there is no teacher that can correct useless or wrong actions. Another motive of difficulty for the learning Animat is the stage setting problem: how can the Animat realize that a particular action, although not directly rewarded, is important because it is a necessary step on the route to the goal? Wilson says that the Animat problem can be summarized as "the problem of incrementally learning multiple disjunctive concepts under payoff." Because of this characterization of the Animat problem he could propose a first investigation using as a test-bed the easily defined *multiplexer problem.* Facing this problem with the BOOLE system, he proved the feasibility of classifier systems as a tool to learn disjunctive concepts under payoff, even if in an environment much simpler than the one a real Animat would probably ever be in. The work done by Wilson is related to ours as far as general background ideas are concerned. In our research we address some of the problems Wilson considered to be of great importance for the development of working Animats: how to control the growth in complexity faced by a learning classifier system that has to solve real world problems, how to coordinate modules and how to use them in a hierarchical structure.

The general ideas presented in our paper are very close to the work presented in the Gofer system [2], but they extend the approach described there in the following respects.

- In the Gofer system, the Tinbergen model is just used as a framework to implicitly describe global robot behavior (modularity, activation etc.). But the model is not explicitly represented in the system. All the behaviors (explore, approach, escape) are incorporated into one single classifier system, whereas in our system each behavior is represented by an individual classifier system. This structure has allowed us to solve and overcome some of the problems Booker reported (e.g., competition between classifiers that realize completely different behaviors, extinction of particular behavioral sequences not relevant in a particular situation, insufficient distinction between coordination messages and action rules).
- Gofer uses a winner-take-all strategy to select appropriate robot behavior. In our system, we use a mediation technique to summarize emitted motor signals as already mentioned above.
- The introduction of hierarchical classifier systems enables the system to distribute models of behavioral sequences over the complete architecture. Instead of forming long sequences of action rules using the bucket-brigade algorithm in a single classifier system, the system reduces the length of related action rules by building individual action sequences and models of their interaction. As discussed also in [20], the bucket-brigade algorithm may lose effectiveness as action sequences grow. Therefore the reduction of length of bucket-brigade chains would make reinforcement and learning faster. A further advantage Wilson cites in his paper is that the hierarchical organization reflects more naturally the real character of animal and human behavioral organization and leads to more powerful mental models of the world.

Another major contribute to the understanding of how to apply learning classifier systems to the Animat problem is the work of Zhou [21]. In his classifier system with memory (CSM) system he addresses the problem of long versus short term memory, i.e., how to use past experience to ease the problem solving activity in novel situations. Zhou's approach is to build a system in which a short and a long term memory are simultaneously present. The short term memory is just the standard set of rules found in every learning classifier system; the long term memory is a set of rule chunks, where every rule chunk represents a generalized version of problem solving expertise acquired in previous problem solving activity. Every time the Animat is presented a problem it starts the learning procedures trying to use long-term experience by means of an appropriate initialization mechanism. Thereafter, the system works as a standard classifier system—except for some minor changes—until an acceptable level of performance has been achieved. It is at this point that a generalizer process takes control and compress the acquired knowledge into a chunk of rules that are memorized for later use in the long term memory.

In his work Zhou does not consider the coordination/cooperation aspects of learning, and the memory is simply filled with chunks of rules, each one usable to solve a set of problems or to initialize similar problem solving activities reducing the learning effort, but with no other hierarchical structure. On the other hand, in our system coordination of learned behaviors has been explicitly introduced, but no explicit long term memory, even if coordination rules at higher level in the hierarchy can be seen as long term memory for action plans actuated with actions taking place at lower levels. We believe that an integration between the two approaches could lead to systems with a still greater capacity to govern high degree of complexity.

Last there is the work of Grefenstette [11], where the problem of learning decision rules for sequential tasks is addressed. Grefenstette's work radically differs from ours in that he does not apply genetic operators to individual rules. Instead he recombines plans, where a plan in his terminology is a set of rules. Also his research goal is different: he investigates the effect that training data have on performance in a target environment.

VI. Simulation Results and Discussion

An important aspect of our work is to study the actual advantages of using such a complex behavioral organization as far as the increased robustness and flexibility of the robot's behavior is concerned. As it is difficult to estimate the changing performance of a robot controlled by various control mechanisms and by the use of internal world models, we had to design experiments in order to evaluate the expected advantages of our particular model of behavioral organization.

We are currently building a real mobile robot equipped with various sensors such as ultrasonic, infrared, and touch sensors [9], [23]. The robot will use our model of behavioral organization in order to control its behavior and to interact with the environment that is unstructured. The general properties of our computational model will enable the robot to relate simple sensor invariants to more complex sensor invariant configurations hence abstracting to higher concepts via generalization and learning.

We believe that, in the long term, the use of a real robot will be mandatory because of the close interdependence between environment and adaptive processes. At the lowest level of these genetics based learning activities, i.e. the correlation between pieces of arbitrary sensory input and useful responses, the notions of disorder and dynamics play an important role, since only through environmental feedback the system assigns an internal *semantics* to the sensory input. Simulations providing structured and predictable environments will never (or only with immense computing efforts) serve this purpose.

But learning cycles using a real robot are time consuming and require an enormous experimental effort. For these reasons, we need simulations to develop a system rapidly and to test our ideas. Later on, the training sets developed during the simulations can be downloaded onto the real robot that can refine them through experimentation and on-line learning. Results of simulations will therefore be used to provide our robot with basic, though raw, behavioral skills.

With this goal in mind, we have built a simulated environment in order to evaluate the performance of our current model. A simulated robot with simple capabilities learns some behavioral patterns and how to coordinate them. A major problem was to design appropriate feedback functions and to incorporate various feedback aspects in the hierarchical organization of the modules to achieve the learning of the desired behavioral patterns. We started with independent classifier systems having individual feedback models each representing the particular task to be learned. These feedback models are directly linked to specific real-world parameters such as distance, temperature, light intensity etc. Another problem was to define the appropriate feedback function for the classifier system located at the second level of the hierarchy, since its payoff cannot be governed by the reward achieved from sensory input directly.

In the following sections we first describe the system and its built-in capabilities and then present some results.

A. Environmental Settings

We designed three sets of experiments to investigate the learning of one, two and three behavioral patterns at the same time, and different coordination techniques to moderate between them. In our experiments we used two kinds of simulated robots (Rob1 and Rob2) living in a two-dimensional (2-D) world. Both robots have a square shape with each type of sensor on each side. Also the movement capability is completely symmetric along the two axis. We cannot therefore talk of "forward" and "backward" or of "left" and "right." All movements will be referred to absolute directions (North–East–South–West). The sensing capabilities of the two robots are:

Rob1's sensors.

- Four light sensors (see Fig. 7) that cover, with overlays, the whole environment; the output of each light sensor is a binary value (0/1) and indicates whether there is light (1) or not (0) in the half space this particular sensor is monitoring; the four bits delivered by the four sensors compose a message that is read by the detectors of the LCS. Messages have a structure as shown in Fig. 8. An example illustrates this: if the light is in the position shown in Fig. 7, only sensors N and E will be activated, and the corresponding message 1100 will be received by detectors of the LCS.
- Four heat sensors, on each of the four sides; they provide the LCS with messages that have the same structure as messages coming from light sensors; the main difference is that they perceive the heat source only when the robot is closer to it than a threshold distance.

Rob2's sensors.

- A set of four light sensors as for Rob1.
- Further, it has sensors to sense information about "food," generating messages having the same structure as the ones coming from the light sensors (i.e., four bit).
- It has sensors to sense information about "predators," generating messages that also have the same structure as the ones coming from the light sensors.

Both Rob1 and Rob2 are allowed to move into eight different directions (see Fig. 9). A four bit message is sent by the LCS to the effectors to cause a robot movement; three bits to specify the direction of turning, and one to specify motion or otherwise.

The learning goals of the two robots were different. The capabilities of Rob1 were tested on a very simple problem;

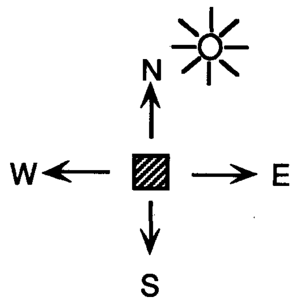


Fig. 7. Robot sensing.

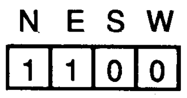


Fig. 8. Structure of a message going from light sensors to detectors.

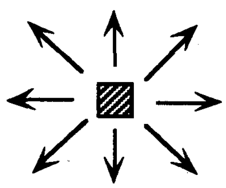


Fig. 9. Possible robot movement directions.

to learn to follow a moving light source. Additionally, we investigated the emergence of structures within the rule base of the classifier system that correspond to (static and dynamic) properties of the real world (experiments described in Section VI-B-1). Thereafter we investigated how Rob1 learned the coordination of two conflicting behavioral patterns. First, it learned to follow a light source while avoiding dangerous hot objects. Second, it learned to follow two independently moving light sources simultaneously. Thus it learned to minimize the average distance to the two light sources. In the context of these two settings, we investigated the properties of different organization principles within the robot controller (monolithic, switch and hierarchical organization of behavioral modules; experiments described in Section VI-B-2). Finally, we studied the learning of three behavioral patterns at the same time and the coordination between them. Therefore, Rob2's task had to be slightly more complicated; it learned to follow the light source, to reach food whenever hungry and available, and to avoid predators (experiments described in section VI-B-3).

B. Details of Experiments

1) Learning to Follow a Moving Light Source: In the first set of experiments we evaluated the performance of Rob1 in accomplishing one of its subtasks: to learn to follow a moving light source. During the experiment the light moves along a circular path (see Fig. 10).

Fig. 11 shows that the learning rate, measured as the distance of the robot to the light source, decreases until the correct behavioral pattern is learned. After 250 cycles

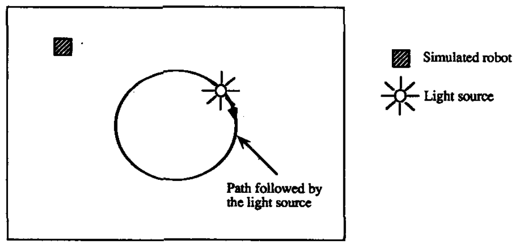


Fig. 10. The initial state when Rob1 is learning to follow a light source.

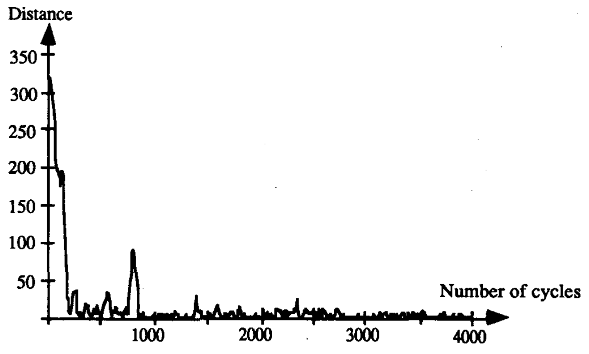


Fig. 11. Distance (in pixel) of the robot to the moving light source.

(150 seconds using three transputers and a population of 300 classifiers) the system already shows a good performance, after 900 cycles we can say it has learned the desired behavioral pattern.

We then ran a set of experiments to test if Rob1 was able to build up internal structures that correspond to features of the external world. Our main interest herein was to understand whether these internal structures could be interpreted as internal representations of the robot controller.[3] By internal world model we understand some internal representation in terms of classifiers that do not directly couple sensing to action. Instead, these classifiers trigger system activities in terms of subsequent rule firing that end up in useful behavioral patterns. The structure and the dynamic interaction of these internal messages (as opposed to external messages) can be considered as a learned model of the external world. To investigate this point we made the following experiments.

- We let the light source have an average speed higher than Rob1's speed.
- We compared the system performance in the case of the light following a periodical path (e.g. circular path) and in the case of the light moving along a random path.
- We compared the system performance when, after the system has learned to trace the light following a circular trajectory, the light changes its trajectory to a rectangular one.

[3]We use the terms *internal representation* or *internal world model* not in their traditional sense as knowledge that is actually present in and available to the robot. Rather we use these terms to refer to internal structures of the learning classifier system that generate some fixed behavioral patterns during the robot-world interaction.

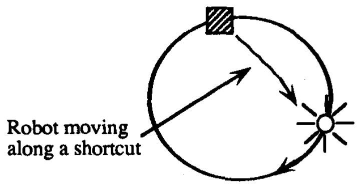


Fig. 12. When the light goes faster than the robot, the robot chooses shortcuts.

Following a faster light source: During this experiment we gave the light source a circular trajectory and let its average speed be higher than the average speed of the robot. As a result, the robot developed anyway the skill to follow the light, but, being too slow to stay in touch, it moved along shortcuts (see Fig. 12). This behavior is a clue, even if not definitive, for the presence of an internal model. In fact, it looks like the system is able to "anticipate" the movements of the light, tracing its movements by means of repeated shortcuts. A simple thought experiment should illustrate this: assuming the shortcuts can be explained without the necessity of anything but stimulus-response rules. In this case, Rob1 has developed a set of simple stimulus-response rules, each one saying: "if position of light is ⟨direction⟩ then move toward ⟨direction⟩." The robot is able to follow a light source nearby. Suddenly the light accelerates: as now the light source is further away from Rob1, the result of applying the same previously cited stimulus-response rules would result in a move towards the light following a direction that, as the light is far away, is no more appropriate to approach the light. Instead of a fixed stimulus-response pair, a different association between a stimulus and a response is formed by means of posting internal messages. Between perceiving the environmental message and releasing an external motor message, subsequent rule firing has taken place and a stimulus is paired with a different response. In this context, we can assume that the robot has learned a model of light source movements.

Following different light paths: In this experiment we taught a first system, Rob1-circular, to follow a light moving along a circular path: the resulting performance is shown in Fig. 13. We then taught a second system, Rob1-random, to follow a light moving along a random path. The result was that the system performance in the first case was higher (see Fig. 13). The higher performance when following the light moving on the circular path can be due to the LCS dynamics: when the LCS proposes a good action, this action, and therefore the rule that proposed it, gets a reward; at the next step the rewarded rule, if the environment has not significantly changed (i.e., the sensory input remains nearly the same), will have a higher probability to fire than before, and therefore to do the right thing again. This is a common situation in a slow changing environment as the one in which the light moves along a circle. On the contrary, in a rapid changing environment (like random light) this useful effect cannot be exploited, making the task a little more difficult.

Following a changing light path: In this experiment we investigated the capacity of a system, which has acquired a particular behavioral pattern, to adapt to a novel situation. In

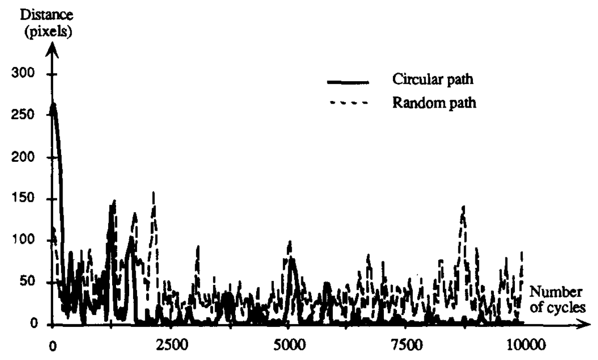


Fig. 13. System performance in case of circular and random path (measured as robot distance from light source).

the experiment the system learned to trace a light moving along a circular trajectory; then the learning algorithms were suddenly stopped and the rule population "frozen"; simultaneously the light trajectory was changed to a rectangular one. We observed a decrease in performance, even if the system was still able to trace the light. A better performance was achieved if the learning algorithms were not stopped. Also the last result cannot be considered to be definitive: the LCS is a dynamic system and rule strengths are updated at every algorithm iteration causing a time changing behavior. This is a desirable property (it allows the system to be adaptive), but makes less cogent the significance of the lower performance level when learning is stopped.

Discussion: As conclusion of this set of experiments, we have seen that our system develops some useful behavioral patterns in simple working environments. Concerning to the observed behavior, it is interesting to note that the robot behavior seems to be more precise than what we could have imagined considering its sensorial capabilities (i.e., it learns to follow a light that moves in eight directions having information only from four sensors). This fact resembles in some way the effect of coarse coding that can be observed in neural networks applications [12], and deserves further investigation.

Further investigation on more complex systems will be necessary to better understand how the system exploits the generation of internal structures and how these can be interpreted as internal representations. Altogether we are still not able to say if an internal model has been developed or not. As a last attempt we compared our results with those obtained when setting the message list length to one. This way, as only messages from the environment are placed onto the message list, the LCS is forced to act as a stimulus-response engine. Again there was no significant performance difference between the two approaches, which proves that no internal models are necessary to explain these observations.

2) Learning to Coordinate Two "Summable"Behaviors: In this experiment we investigated how Rob1 could manage to learn two conflicting, but "summable" behavioral patterns. We say two behavioral patterns are summable if, when simultaneously active, they propose actions that, even if different, can be combined into a resulting action that partially satisfies the requirements of both behaviors. As outlined before, in

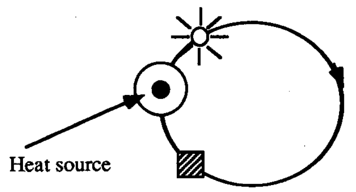


Fig. 14. The environment with the heat source.

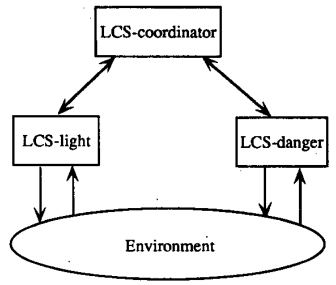


Fig. 15. Architecture of the system used in the heat and light experiment.

this section we focus on two different learning environments, one heat and light environment, where the task is to learn to coordinate two concurrently operating classifier systems, and a two light environment, where the objective was to study the use of different coordination mechanisms between concurrently operating classifier systems.

Heat and light environment: In this experiment Rob1 had to learn to follow a moving light source and to learn to avoid dangerous (hot) objects. To make things harder we positioned the hot object along the light trajectory (see Fig. 14).

As we were interested in evaluating the performance of a hierarchical architecture, we designed a two level hierarchy where a LCS, called LCS-light, was specialized in learning the light following behavior, a second LCS, called LCS-danger, was specialized in learning to avoid hot objects and a third LCS, called LCS-coordinator, was specialized in learning the coordination policy.

As actions proposed by LCS-light and LCS-danger are vectors (they represent a move in a 2-D space), they can be summed to produce the final action. Therefore, the coordinator task is to learn to assign appropriate weights to the actions (vectors) proposed by the two low-level classifiers.

In Fig. 15 the system architecture is shown (equivalent to the one presented in Fig. 6). LCS-light, LCS-danger, and LCS-coordinator are implemented as processes running asynchronously on different nodes of a transputer system [8].

The experiment can be ideally divided into two parts: when the light is far away from the hot source, the behavior of Rob1 is the same as in the light following experiment, while when the light source is close to the hot object then Rob1 should move around the hot object continuing to follow the light. LCS-coordinator becomes active only in this second situation.

When the two low level LCS (LCS-light and LCS-danger) are activated simultaneously, the resulting action is a weighted average of the two proposed actions, with weights given by

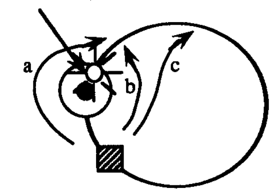


Fig. 16. Robot avoiding the heat source.

the strengths of the two classifiers that proposed the actions (one belonging to LCS-light and the other to LCS-danger) and the excitation level of the classifier system that proposed the action (i.e., the excitation level of LCS-light and LCS-danger, respectively). The procedure is the following: each time LCS-light or LCS-danger post an action message, they also send it to LCS-coordinator. LCS-coordinator just monitors the messages it is receiving. When a situation occurs in which both LCS-light and LCS-danger try to perform an action on the environment, LCS-coordinator reacts sending back to the two classifier systems a message containing information that causes the receiving classifier systems to increase or decrease their excitation level. This way LCS-coordinator can control the cooperation between LCS-light and LCS-danger. LCS-coordinator, after sending messages, receives a reward,[4] which gives information about the usefulness of the action actually performed. In this way LCS-coordinator learns how to control LCS-light and LCS-danger, because it has direct feedback on its own actions and can use it to evaluate its own rules.

Experiments with this architecture were very encouraging. The observed behavior was the desired one, i.e., the robot followed the light until it approached the heat source. Then it chose between two different behavioral patterns: it just turned around the heat source (case a and b in Fig. 16) or it stopped waiting in front of the heat source and, when the light had moved away from the hot region, began to follow it, again (case c in Fig. 16).

Two lights environment: In the preceding experiment we tested the feasibility of a simple hierarchy of LCSs. We now turn our attention to the assessment of its utility, with respect to a more traditional architecture (see the *monolithic architecture* below) in which a single LCS learns both behaviors. We needed an experimental environment in which performance was easy to calculate: we decided to let Rob1 learn to follow two independently moving light sources. (Rob1 distinguishes the two light sources by their different color.) In this environment the performance index (as always specified by the proportion of correct moves to the total number of moves) can be used for analysis since the two stimuli are always present.[5]

We compared the following architectural organizations.

- *Monolithic* (where a single LCS had to learn the three behavioral patterns contemporaneously).

[4] LCS-coordinator is rewarded whenever the resulting action is the correct one.

[5] Remember that in the heat and light environment the heat source was perceived only when Rob1 distance to it was below a given threshold.

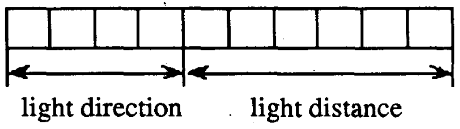


Fig. 17. Structure of a message going to LCS-light in the vectorial and hierarchical architectures (messages going to the monolithic architecture are a concatenation of two messages like the one in the figure, one for each light to be followed).

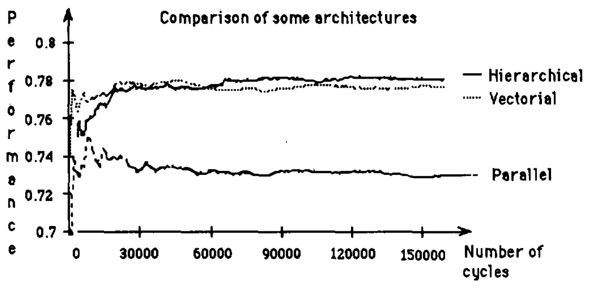


Fig. 18. Comparison between the monolithic, vectorial and hierarchical architectures.

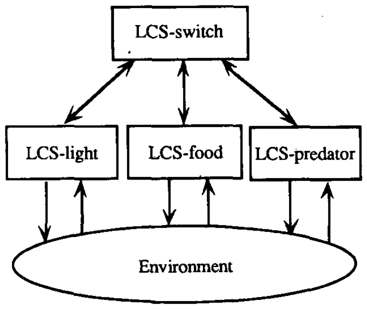
- *Vectorial* (where the coordinator was not a LCS but a procedure that computed a vectorial sum of the two proposed actions).
- *Hierarchical* (the same structure as used in the preceding "heat and light" environment).

In this task we have two sets of four light sensors, each set being sensitive to only one of the two colors. Sensors not only communicate to the LCS the light position, but also the light distance. This information is necessary because Rob1 must learn from sensory data which light is the closest and which one is the most distant. The structure of messages is therefore different from the one of Fig. 8; moreover, we have different environmental messages for the three architectures investigated: very long messages in the monolithic architecture, shorter messages in the hierarchical and vectorial architectures (see Fig. 17). As in the monolithic architecture messages are longer, the learning task is more difficult (the search space is bigger, as it grows with 3^k, where k is the classifier length), and we expect therefore the monolithic architecture performance to be the worst.

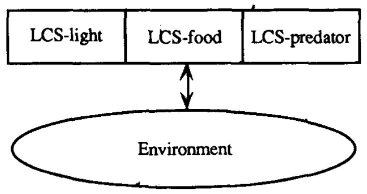
In Fig. 18 we report results obtained comparing the three architectures. Our hypothesis about the lower performance level of the monolithic architecture has been confirmed (consider that the dimension of the search space in the monolithic architecture case is 3^{60}; in fact, environmental messages are 20 bits long, and therefore rules that are composed of two conditions and one action are 60 bits long; in the other two architectures environmental messages are only 10 bits long, and therefore rules are 30 bits long and the search space[6] is 3^{30}).

It is also interesting to note that the vectorial architecture, as it does not have to learn the coordination policy, learns quicker, but after 60 thousands of cycles the coordination policy learned by the coordinator becomes slightly more effective.

[6]The dimension of the search space of the coordinator in the hierarchical architecture is much smaller (3^{12}, because each low level LCS sends a 4 bit message) and is therefore not considered.



(a)



(b)

Fig. 19. (a) The switch architecture. (b) The monolithic hierarchical architecture.

3) Learning to Coordinate Three Different Behaviors: This experiment was devoted to evaluate a different kind of coordination in which the coordinator must learn to switch between three behavioral patterns. The three basic low level LCSs are called LCS-light, LCS-food and LCS-predator. As the three activities to be learned cannot be performed contemporaneously,[7] the coordinator has to learn a "switching" policy, i.e., to which low level LCS to give control to, when more than one of them is active (i.e., proposes an action). We call the LCS implementing the coordinator LCS-switch. The structure of the learning system is shown in Fig. 19(a).

Messages received by LCS-switch are three bits messages, each bit saying whether the corresponding low level LCS proposes an action (bit set to 1) or not (bit set to 0). The learning task for LCS-switch was to give the highest priority to LCS-predator in case this behavior is active, and to choose LCS-food whenever only LCS-light and LCS-food are active.

It is interesting to note that, when using hierarchies of LCSs, it is possible to define many reward strategies. In our experiments we tested two of them: in the first one we let all the LCSs learn contemporaneously, in the second one we first reward low level LCSs, then we "freeze" the learning algorithms of low level LCSs and start rewarding the LCS-switch.

As with the preceding "two lights environment," we have done experiments to compare the monolithic (see Fig. 19(b)) and the hierarchical architectures (see Fig. 19(a)). In both cases the single behavior performance refers only to moves done when the behavior under consideration was active, while

[7]And it does not make sense to "sum" them. This can be considered an extreme case of summable behaviors, where weights are 0 or 1.

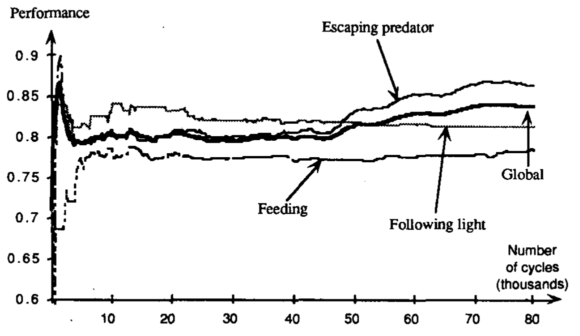


Fig. 20. Performance using the monolithic architecture. Searching food has a lower performance than other behaviors because it is more difficult to learn. The whole experiment lasted approximately 8 h.

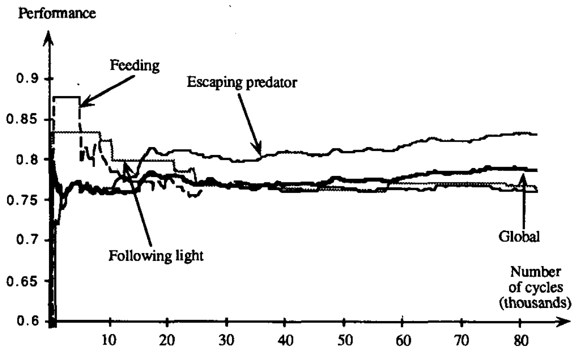


Fig. 21. Performance using the switch architecture. The whole experiment lasted approximately 8 h.

global performance considers all the moves (i.e., if LCS-light has learned very well and LCS-predator has not, then the global behavior performance will settle somewhere between them).

Experiments with the monolithic architecture: In this case a single LCS[8] was used. An environmental message is equivalent to the concatenation of the three messages coming from the three groups of sensors. A rule in the monolithic architecture is therefore 36 bits long.

It is important to note that in the monolithic architecture coordination is not achieved by an explicit LCS (as in the hierarchical architecture), but it is the result of the learning process of the unique LCS implementing the three low level behaviors. It is also interesting to note that the performance level of the escaping behavioral pattern settled at a higher performance value (see Fig. 20 and 21). Escaping predator is an easier task because the number of correct moves on the number of possible moves is the greatest (in the discrete 2-D world in which our simulated robots live there are more departing directions from one point than approaching directions to the same point).

We report in Figs. 20 and 21 the performance of the simulated robot, for both the monolithic and the switch architectures.

[8]The LCS was parallelized using four T800 transputers [8].

Experiments with the switch architecture: The experiment was run to see whether the use of a hierarchical architecture improved the performance level. As in the "two lights environment," the idea is that each single LCS, having shorter classifiers, has a smaller search space and that therefore the overall learning task could be easier. The presence of the coordinator should not create efficiency problems, due to the easiness of its learning task: it must learn to choose which of the three lower level classifier systems to give priority, when they contemporaneously propose an action.

Low level LCSs send 5 bit messages: these messages are composed of 4 bits, which propose an action and of one bit going to the coordinator and saying if the sending LCS is proposing an action or not. The coordinator therefore receives three bit messages, indicating which low level classifiers are active, and produces an action that determines which low level LCS should take control of the robot; then the action that was proposed by the LCS, which received the control, is performed and a reward is assigned to the system. In a first experiment the reward function considers the whole learning system as a black box, i.e., we did not use the actual behavior of single classifier systems to distribute rewards (but we observed them in order to monitor performance). This policy makes the learning task more difficult, especially in the case of hierarchical architectures such as the switch architecture, because there can be situations in which a correct action is the result of two wrong messages (e.g., the switch chooses to give control to the wrong low level LCS that in turn proposes a wrong move that, in this context, results to be the right one) or situations in which a LCS gets a punishment because of the mistake another LCS did. Nevertheless this way of giving rewards is an interesting one because it does not require accessibility to all the internal modules of the system and is much more plausible from an ethological point of view. Rule length in low level classifier systems is 12 bits, whereas in the switch it is 9 bits.

Results obtained using the switch architecture show that the relative performance of the three behavioral patterns is the same as with the monolithic architecture. The absolute performance is a little lower than with the monolithic architecture: this is probably due to the noise in the reward function (as explained before).

The performance achievable by the hierarchical architecture improves when we use a two phase reward policy. In the first phase (see Fig. 22), we reward every low level behavioral module its own behavior. This activity can go on in parallel (in three simulated worlds in which there is only one sensory stimulus): a module learns to follow the light, another one to find the food and the last one to escape the predator. When they all have achieved a good performance level, learning is stopped (commencing at cycle 30000 in Fig. 22) and the learning phase of LCS-switch starts (i.e., in this phase it is the switch that learns, while low level LCSs are "frozen"). After cycle 42000 we froze the whole system. Using this reward policy, we obtained an improvement in the system performance, both regarding the performance level achieved and the number of cycles required to achieve it. It is clear what importance good reward methodologies have when building

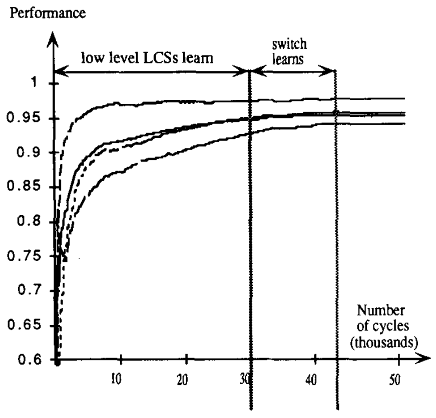


Fig. 22. Performance using the switch architecture and a two phases reward policy. Ten thousand cycles required 1-h computing time.

more complex systems.

It will be the subject of future research to study these reward methodologies, i.e., how to reward and how to organize the learning process (all tasks are learned together, low level LCSs learn first then learns the coordination level or other).

VII. Conclusion

We have presented some results of a hierarchical and parallel model of behavioral organization in a simulated robot. Starting from a short survey of genetics-based machine learning techniques and behavior-based robotics, we have outlined the current implementation of our computational model and we have compared our approach with related work done by other researchers. The results of simple experiments designed and carried out to evaluate the current implementation of the learning system have been discussed. The results are promising and seem to indicate that the use of explicit coordination learning to combine primitive behaviors hence improving the systems adaptability can be appropriate.

Further work is going on in the direction of a better understanding of the learning properties of our system. We are also building in the system new features to make the implementation closer to the Tinbergen's model. Experiments on a real robot in order to test the developed model in a real environment are beginning [9]. Another important aspect to be investigated will be how to memorize, and use, past experience. Two solutions seem at this moment to be feasible: the use of explicit memory structures, as in the work of Zhou [21], or the insertion of implicit memory structures in the system architecture.

Acknowledgment

The authors would like to thank Andrea Bonarini, Marco Colombetti, and Vittorio Maniezzo for their useful suggestions on earlier versions of this paper. The authors would also like to thank Mukesh Patel for his careful proofreading and providing useful comments.

References

[1] R. C. Arkin, "Neuroscience in Motion, The application of schema theory to mobile robotics," in *Visuomotor Coordination: Amphibians, Comparisons, Models, and Robots,* E. Ewert and M. Arbib, Eds. New York: Plenum, 1987, pp. 649–671.
[2] L. Booker, "Classifier systems that learn internal world models," *Machine Learning,* vol. 3, no. 3, pp. 161–192, 1988.
[3] L. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial Intell.,* vol. 40, pp. 235–282, 1989.
[4] R. A. Brooks, "A robust layered control system for a mobile robot," Artificial Intell. Memo 864, MIT, Cambridge, MA, 1985.
[5] ——, "Achieving artificial intelligence through building robots," Artificial Intell. Memo 899, MIT, Cambridge, MA, 1986.
[6] R. A. Brooks and P. Maes, "Learning to coordinate behaviors," in *Proc. AAAI 90,* 1990, USA.
[7] R. A. Brooks, "Artificial life and real robots," to appear in *Proc. 1st Eur. Conf. Artificial Life,* Dec. 1991, Paris, France.
[8] M. Dorigo, "Using transputers to increase speed and flexibility of genetics-based machine learning systems," *Microprocessing and Microprogramming J.,* vol. 34, pp. 147–152, 1992.
[9] M. Dorigo, "Alecys and the AutonoMouse: Learning to control a real robot by distributed classifer systems," Rapporto Tecnico no. 92–011, Dip. Elett. e Inf., Politecnico di Milano, Milano, Italy, 1992.
[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* New York: Addison-Wesley, 1989.
[11] J. J. Grefenstette, C. L. Ramsey, and A. C. Schulz, "Learning sequential decision rules using simulation models and competition," *Machine Learning,* vol. 5, pp. 355–382, 1990.
[12] G. E. Hinton, J. H. McClelland, and D. E. Rumelhart, "Distributed representations," in *Parallel Distributed Processing,* vol. 1. Cambridge, MA: MIT Press, 1986, pp. 77–109.
[13] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press, 1975.
[14] J. H. Holland, "Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems," in *Machine Learning II,* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. San Mateo, CA: Morgan Kaufmann, 1986.
[15] O. Holland and M. Snaith, "The use of neural modelling in the development of autonomous mobile robots," Tech. Rep. Technol. Appl. Group, Alnwick, UK, 1990.
[16] P. Maes, "How to do the right thing," *Connection Sci.,* vol. 1, no. 3, 1990.
[17] U. Schnepf, "Robot ethology: A proposal for the research into intelligent autonomous systems," in *Proc. Int. Conf. Simulation of Adaptive Behavior: From Animals to Animats* (SAB-90), Sept. 1990, Paris, France.
[18] N. Tinbergen, *The Study of Instincts.* Oxford, UK: Oxford Univ. Press, 1966.
[19] S. Wilson, "Classifier systems and the animat problem," *Machine Learning,* vol. 2, no. 3, pp. 199–228, 1987.
[20] S. Wilson, "Hierarchical credit allocation in a classifier system," in *Proc. 10th Int. Joint Conf. Artificial Intell.,* Aug. 23–28, 1987, Milan, Italy, pp. 217–220.
[21] H. H. Zhou, "CSM: A computational model of cumulative learning," *Machine Learning,* vol. 5, no. 4, pp. 383–406, 1990.
[22] A. Bertoni and M. Dorigo, "Implicit parallelism in genetic algorithms," to appear in *Artificial Intell.,* 1993.
[23] M. Colombetti and M. Dorigo, "Robot shaping: Developing situated agents through learning," Rapporto Tecnico no. 92–040, Int., Comput. Sci. Inst., Berkeley,. CA, 1992.

Marco Dorigo (M'92) was born in Milan, Italy, in 1961. He received the Laurea (Master of Technology) in industrial technologies engineering, in 1986 and the Ph.D. in electronic engineering of Information and Systems in 1992 from the Politecnico di Milano, Milan, Italy.

He currently holds a postdoctoral position at the International Computer Science Institute, Berkeley, CA. He is a member of the Politecnico di Milano Artificial Intelligence and Robotics Project. He took part to several CEC ESPRIT Projects, and National research projects. His research interests are in the field of machine learning, and in particular of genetic algorithms applied to learning, optimization, and control.

Dr. Dorigo is a member of the Italian Association for Artificial Intelligence (AI*IA) and of the IEEE SMC Society.

Uwe Schnepf was born in Mannheim, Germany, on September 25, 1960. He received his Diplom in mechanical engineering from the University of Kaiserslautern, Germany, in 1987, and the M.Sc. in information technology and knowledge-based systems at the Department of Artificial Intelligence at the University of Edinburgh, Scotland, in 1988.

Since 1989 he has been working at the Artificial Intelligence Research Division of the German National Research Center for Computer Science (GMD) at Sankt Augustin, Germany. His research interests are in the area of behavior-based robotics and machine learning.

Mr. Schnepf is a member of the German Association of Engineers (VDI) and the German Association for Computer Science (GI).