Genetics-based Machine Learning and Behaviour Based Robotics: A New Synthesis

Genetics-based Machine Learning and Behavior Based Robotics:
A New Synthesis, Marco Dorigo, Uwe Schnepf,
IEEE Transactions on System, MAn, and Cybernetics,
23, 1, 141-154, January 1993

Dean Carpenter

Overview

Robots should be able to learn how to behave in a real-world environment

Knowledge based and symbol manipulative AI systems are not flexible enough. Behavior based systems may be a better approach.

Natural Systems have learned to adapt, and this led to neural learning, which is flexible and powerful

The paper deals with genetic machine learning and behavior based robotics

The layout of a genetic learning machine

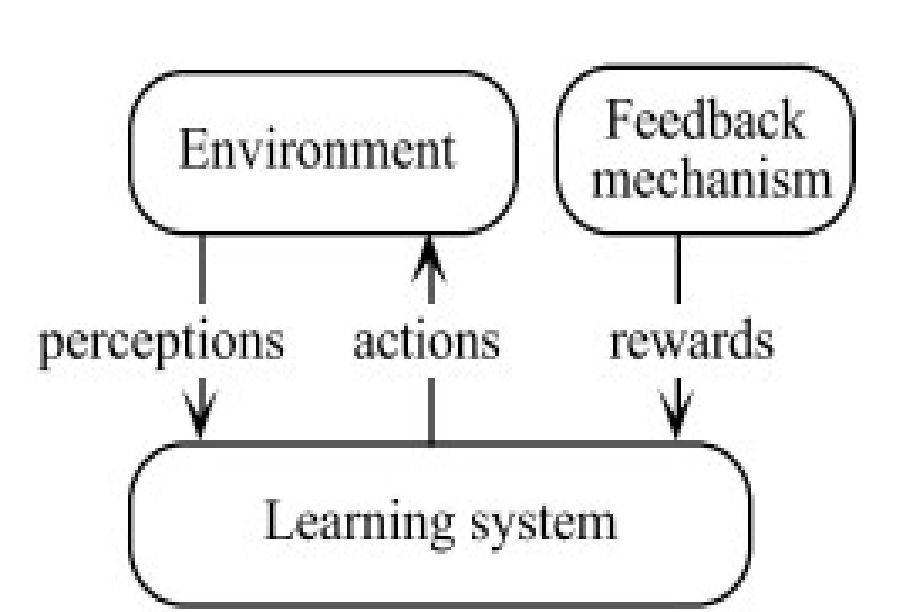

Fig.1 - A general reinforcement learning model.

Genetic Setup

- Rules are strings of symbols over a three-valued alphabet (A ={0,1,*}) with a condition→action format (in their 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.

Structure of the System

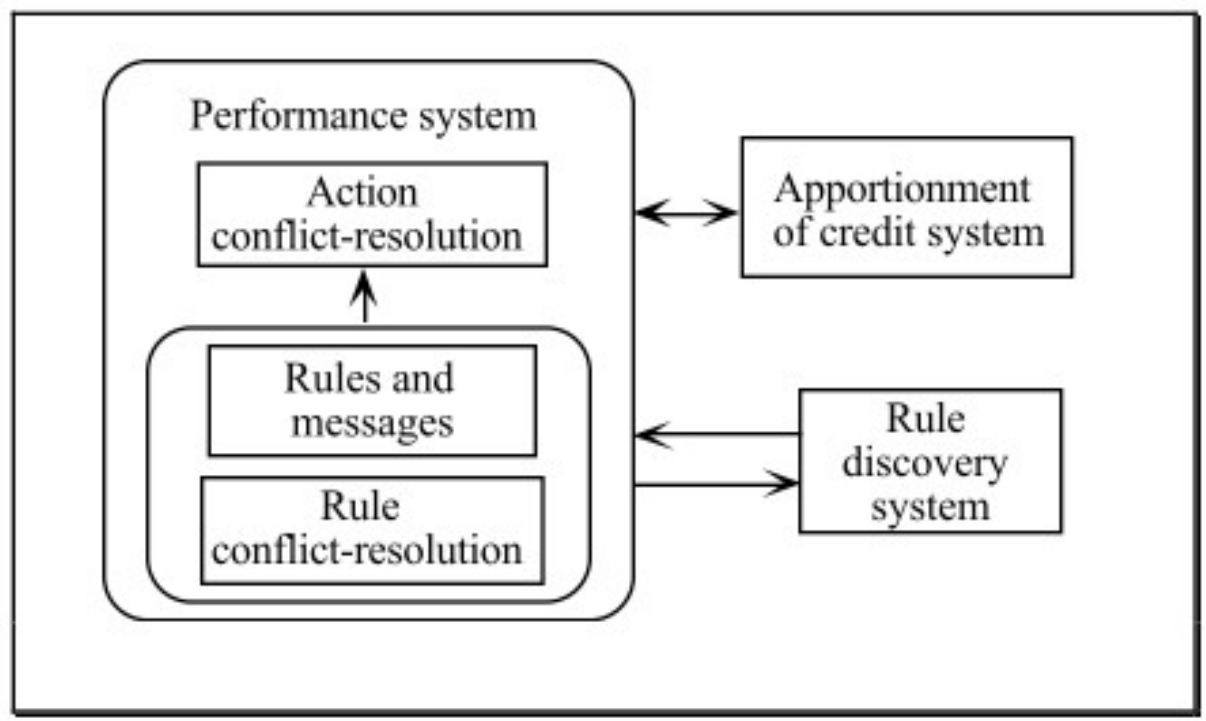

Fig.2 - Structure of a GBML system.

Performance System

- 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.

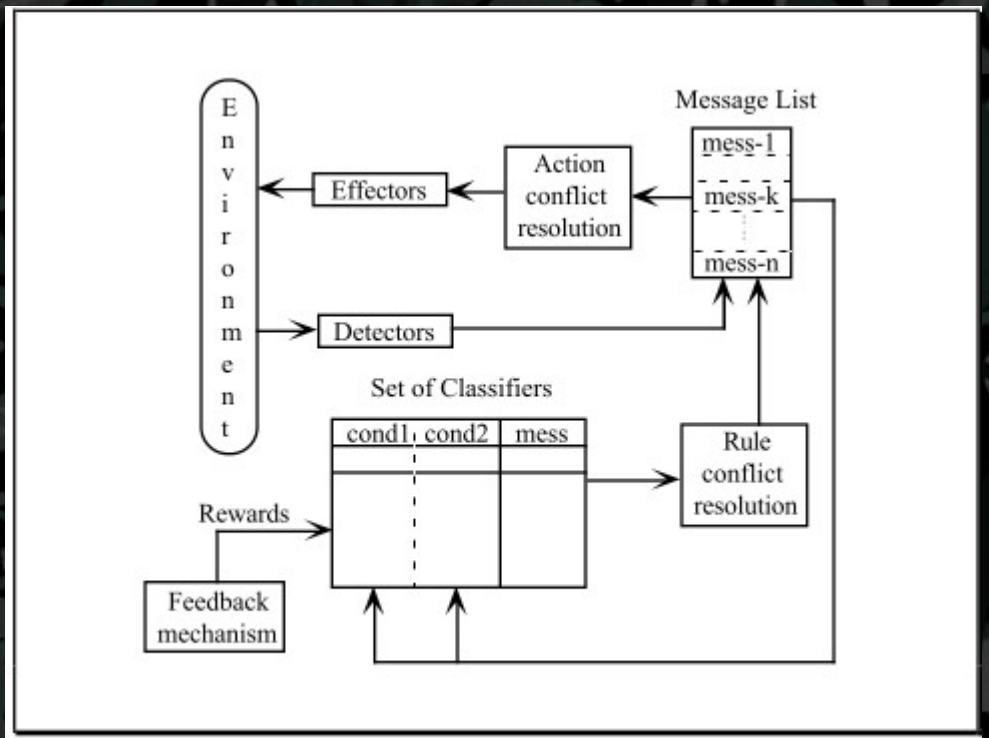

Fig.3 - The performance system.

Terminology

- A classifier (rule) is a string composed of three chromosomes, two chromosomes being the condition part, 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,*}.

Example Classifier

Condition Condition Action

1;011->010

If the message matches both conditions, then the action part is appended to the message stream.

Overview of the algorithm

The algorithm works by feeding the messages through the classifiers in order to get an action output. Depending on the results of the action, the system is either reinforced or punished. It will weight the different classifiers depending on their involvement in the end action. They are then recombined in order to preserve the critical classifiers that lead to the proper output and change the classifiers that lead to improper output.

Behavior Based Learning

Behavior based learning is based on the assumption that cognition arises from trying to impose order on a dynamically changing unstructured environment. The structures it develops are the foundation of high-level thought and action.

These structures did not exist in early life, but developed over time. They are trying to mimic this process in order to achieve robotic intelligence.

Most approaches to this problem have been very structured and engineered. They believe that such attempts are doomed to failure, since they can be well-designed for a particular situation, but a general solution has not been found.

Instinct Centers

They operate under the Tinbergen model of animal behavior , which his 'Instinct centers' which get activated, each of which I composed of finer grained behavior sequences. At any level, only the center that is the most active can activate the levels below it.

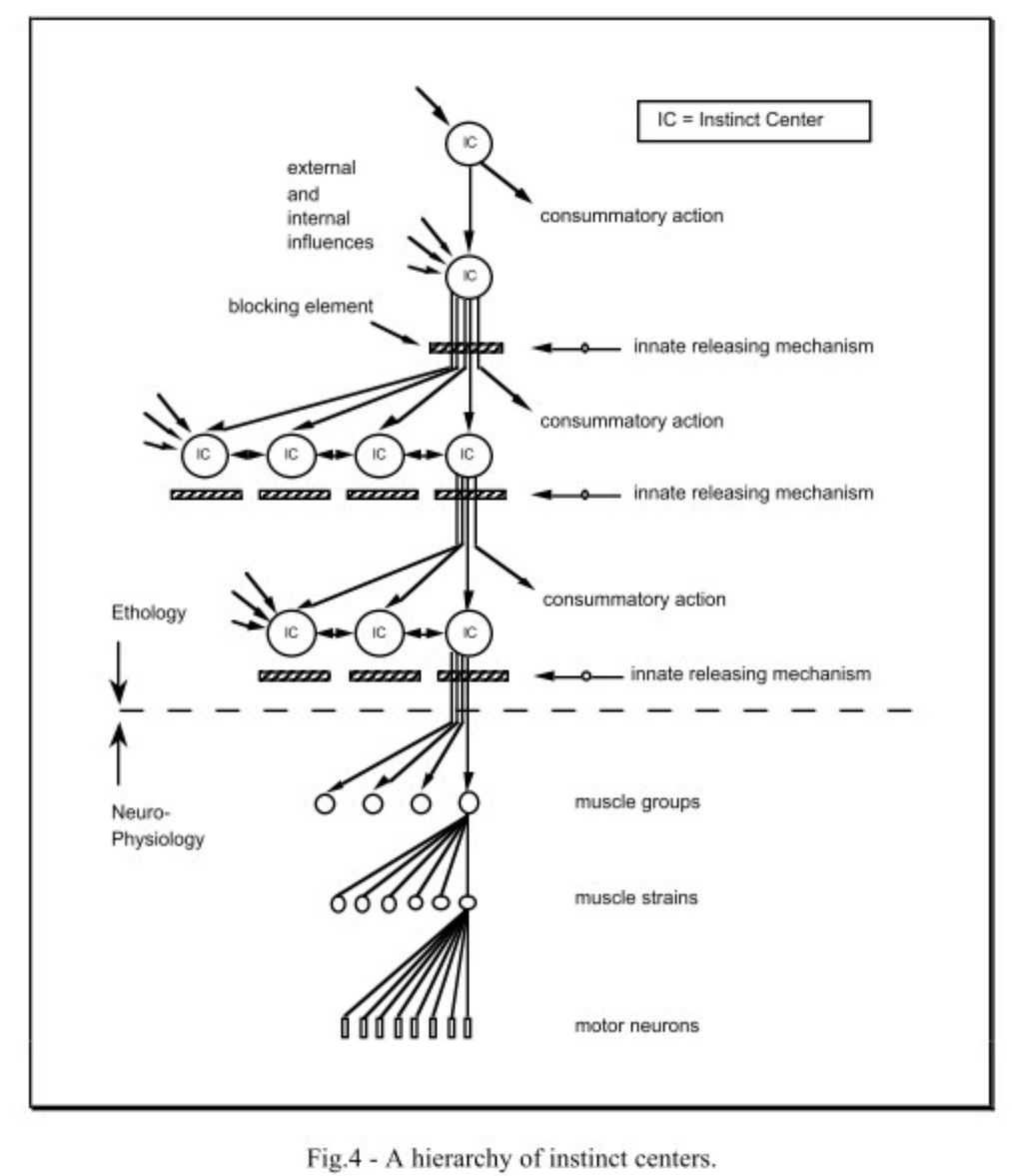

Fig.4 - A hierarchy of instinct centers.

The Complete Model

There are many classifier systems running in parallel. Each classifier learns a single task, and the system as a whole learns to coordinate the tasks. Low level classifiers have direct access to the robots sensors and motors, and high level classifiers operate on lower-level classifiers.

The classifiers are added if the robot encounters a novel situation. The weighted sum of the outputs of the classifiers are used to determine the actual motor outputs

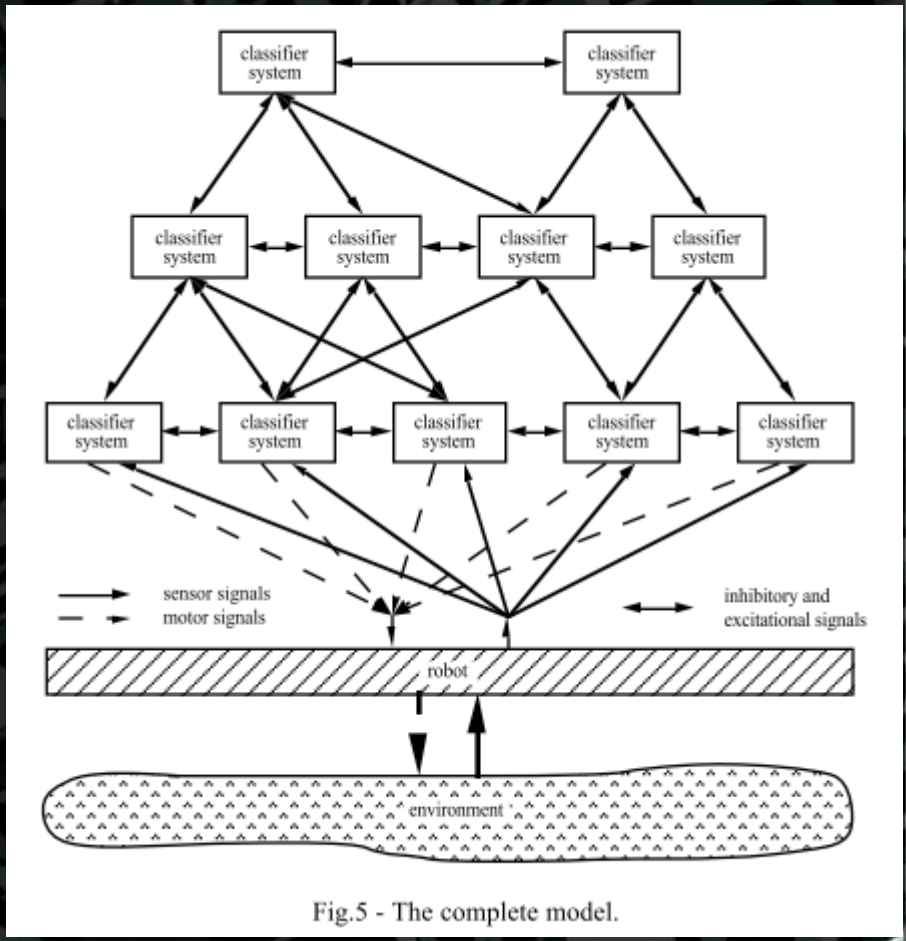

Fig.5 - The complete model.

Simulation Vs. Testing

A system like this needs to be tested, and that test can be done via simulation or by using an actual robot.

A simulation is much faster, but the sensor input is dry and you have a structured environment, which is contrary to their goal.

A robot allows real-world situations to be explored but the testing is much slower. They maintain that real-world interactions are key to developing a working system

They settle on initial simulation and later testing on a real robot

Rob1

Omnidirectional Movement

Four light sensors, each returning 0 or 1

Four heat sensors, each returning 0 or 1

4bit output to specify motion

Designed to learn how to follow light, then learn how to avoid hot objects, then learn how to reconcile contradictory inputs, such as following a light while avoiding a hot object, or following two lights

Rob2

Omnidirectional Movement

Four light sensors, each returning 0 or 1

Food sensor, with input matching the light sensor

Predator Sensor, with input matching the light sensor

4bit output to specify motion

It had to follow 3 directives at the same time; follow light, find and eat food, avoid predators

Following a light source

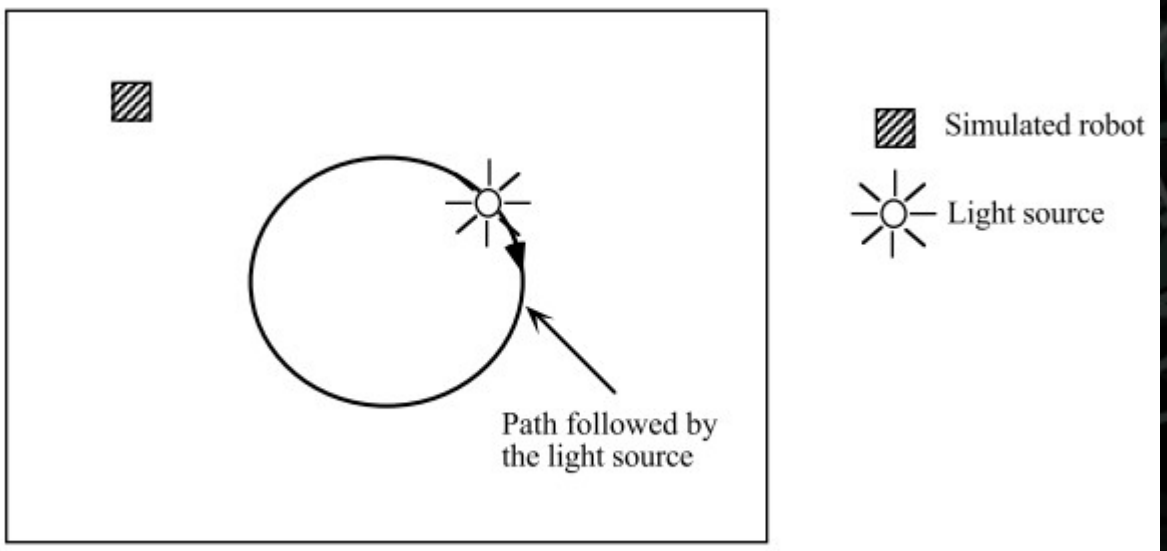

Fig.10 - The initial state when Rob1 is learning to follow a light source.

The robot was simulated to follow a light source that was circling.

After 250 cycles, the robot had good performance, and learned the system by 900 cycles.

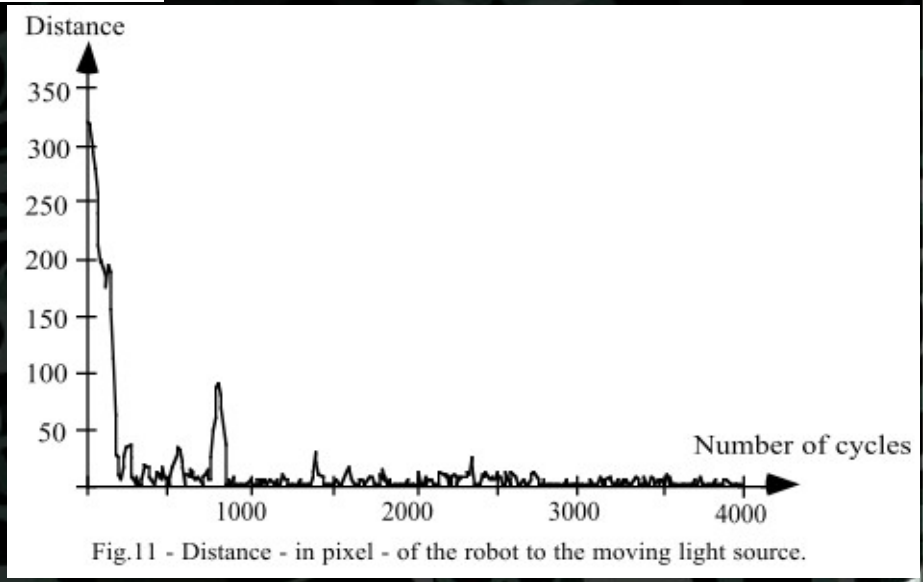

Fig.11 - Distance - in pixel - of the robot to the moving light source.

Testing the internal model

In order to verify that the robot has an internal world model, they performed variations of the experiment to show that it was doing more than coupling inputs to outputs.

They did three experiments to test this:

First, they made the light move faster than the robot.

Second, they made the light move on a random path instead of a circle

Third, after the robot learned the circular path, they changed the path to a rectangle

Faster Light

When they adjusted the speed of the light so it was faster than the robot, the robot started taking shortcuts. This implies an internal model because if it was operating off basic sensor-motor mapping, it will try to follow the light directly. The fact that it can take a shortcut shows that it has enough awareness of the situation to react to it.

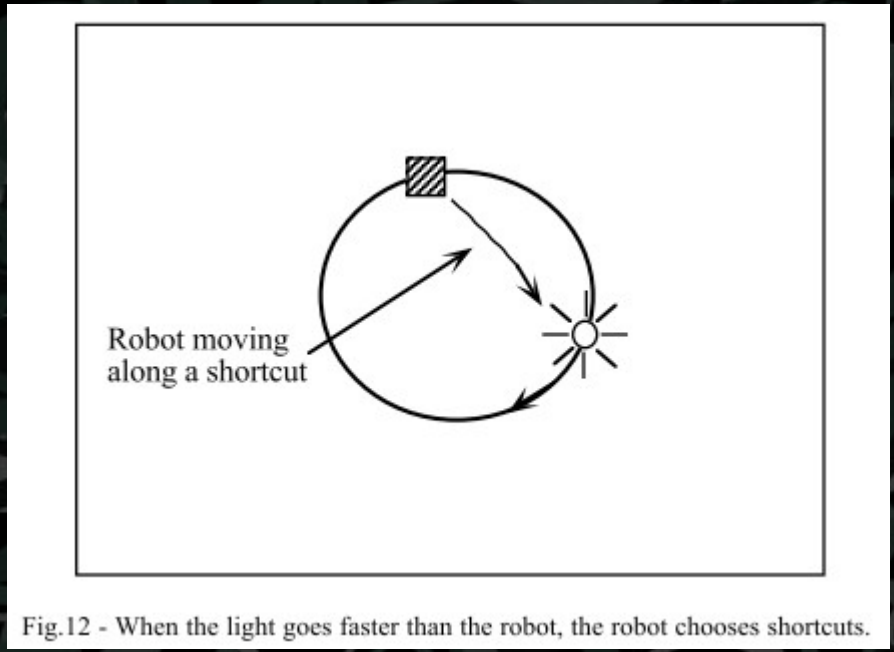

Fig.12 - When the light goes faster than the robot, the robot chooses shortcuts.

Erratic Path

The robot had a harder time following a random light source than the circular one. The reason proposed is that in a a slowly changing system, positive actions have more time to be reinforced. This is the case with the circular path, but that cannot be exploited with a random path, so learning is more difficult.

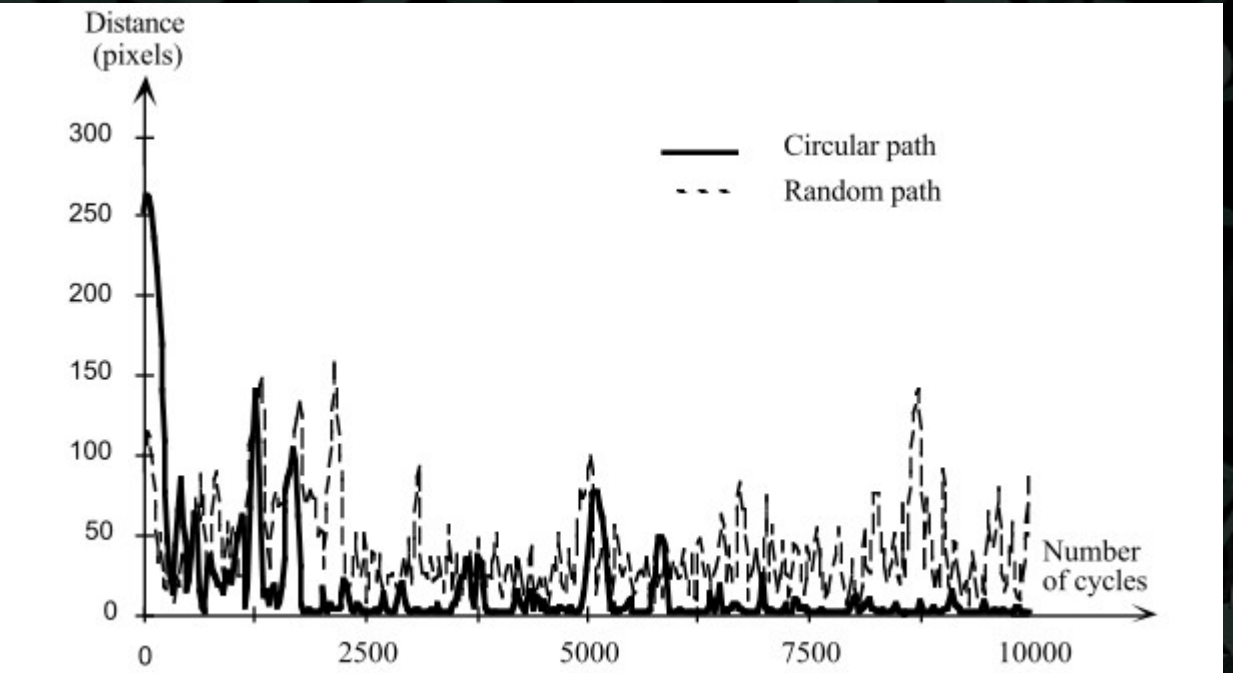

Fig.13 - System performance in case of circular and random path (measured as robot distance from light source).

Rectangular Path

When they froze the learning algorithm and changed the shape of the path, they saw a performance decrease. They achieved better results when they left the learning algorithm in place. They do not consider this definitive because the learning system is a dynamic structure, and freezing it can end up in a suboptimal configuration

Discussion of the first experiment

The robot behavior appears to be more precise than would be expected, considering the output is more fine-grained than the input.

Whether an internal model is present or not is not certain. They tries setting the message length to one, forcing it to act as a input-< output mapper, and there was no significant difference.

More complex systems are needed to figure out if an internal model can be present

Summable actions

The next experiment they tested was to have two inputs that needed to be summed together to get the correct behavior, by avoiding a heat source while following a light, or minimizing the distance between two lights.

These are considered summable because they can both be simultaneously active, and the results of each can be summed to determine the correct course of action

The Light-Heat source setup

For this experiment, they add a heat source to the setup that the robot must avoid. The heat source is placed on the light's path to make it more difficult for the robot.

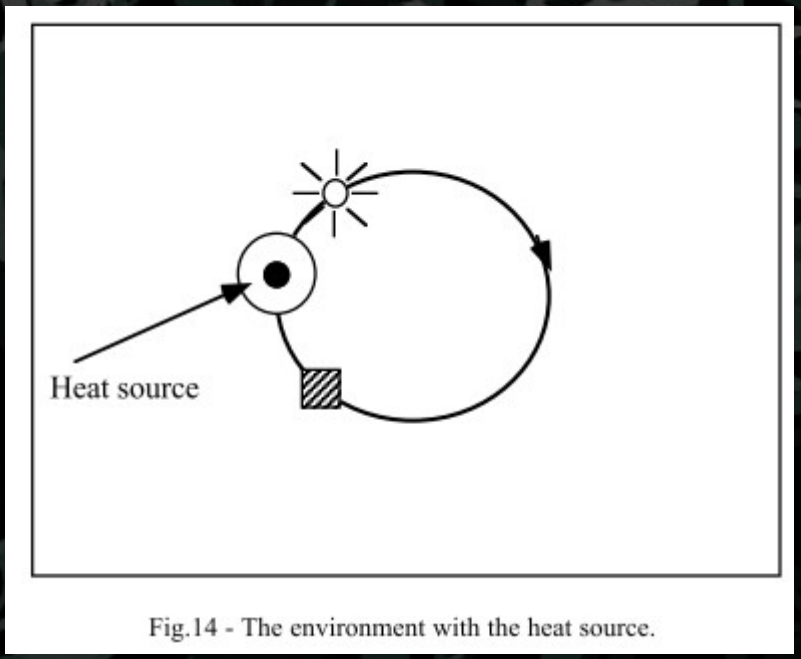

Fig.14 - The environment with the heat source.

Light-Heat architecture

The system is designed with two subsystems to handle the heat avoidance and the light following, which operate in parallel, and a coordinator to combine the two outputs in a single action.

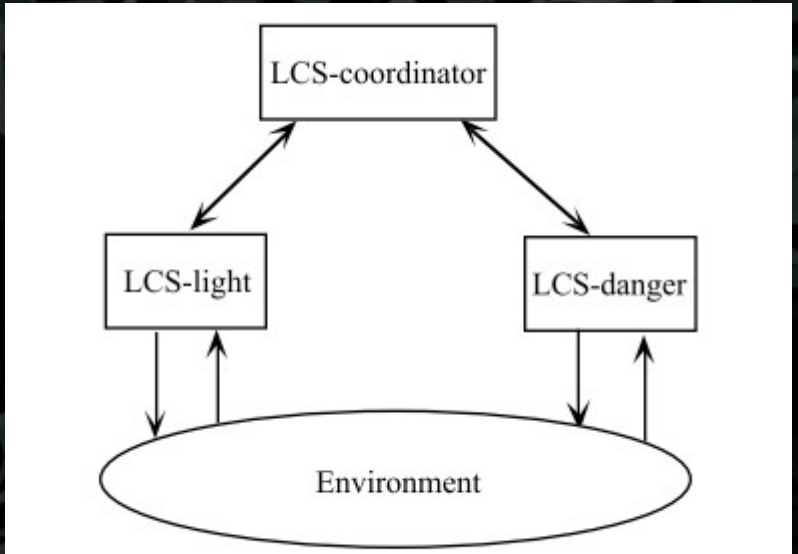

Fig.15 - Architecture of the system used in the heat and light experiment.

Results

The results of this experiment were very promising. The robot displayed the desired behavior; it would follow the light in a circle until it got to the heat source, then it would either go around the heat source or wait until the light is past it and resume following it.

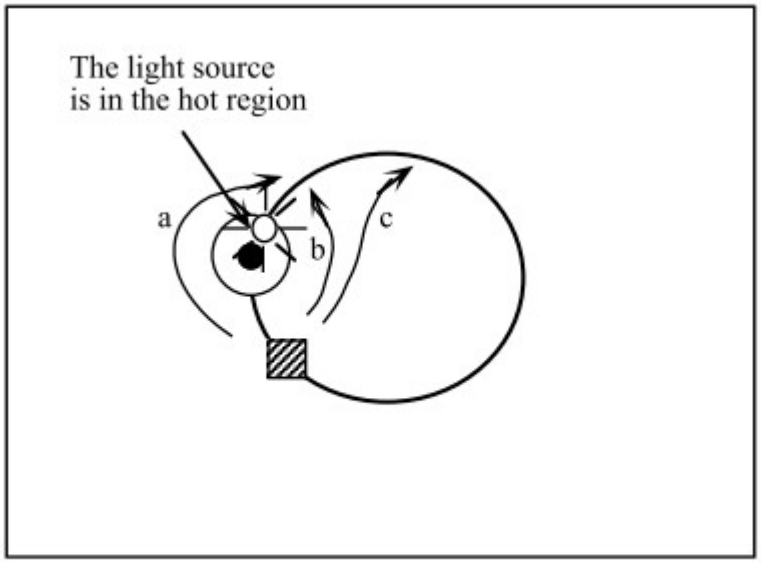

Fig.16 - Robot avoiding the heat source.

Two Lights

The setup for the two lights system in different. The robot is equipped with two sets of the light sensors, each of which can only see one light. The sensors will return both the direction and the distance to both lights, so the robot can try to minimize the distance to both.

Part of this experiment is to try different problem architectures. There are flat, vectorial, and hierarchical.

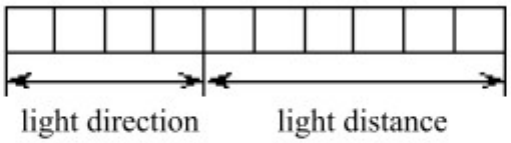

Fig.17 - Structure of a message going to LCS-light in the vectorial and hierarchical architectures (messages going to the flat architecture are a concatenation of two messages like the one in the figure, one for each light to be followed).

Problem Structures

Flat- The entire system has to be learned by a single LCS

Vectorial- Both inputs are learned by a separate LCS, then the results are combined with a Vectorial unit, instead of a LCS.

Hierarchical- this is the same structure that was used in the heat-light problem space.

Results

The flat architecture performed poorly because the number of possible inputs was exponentially large. The vectorial system performed well, especially initially, since the system did not have to learn how to combine the inputs, like the hierarchical structure learned a slightly more efficient method in the end.

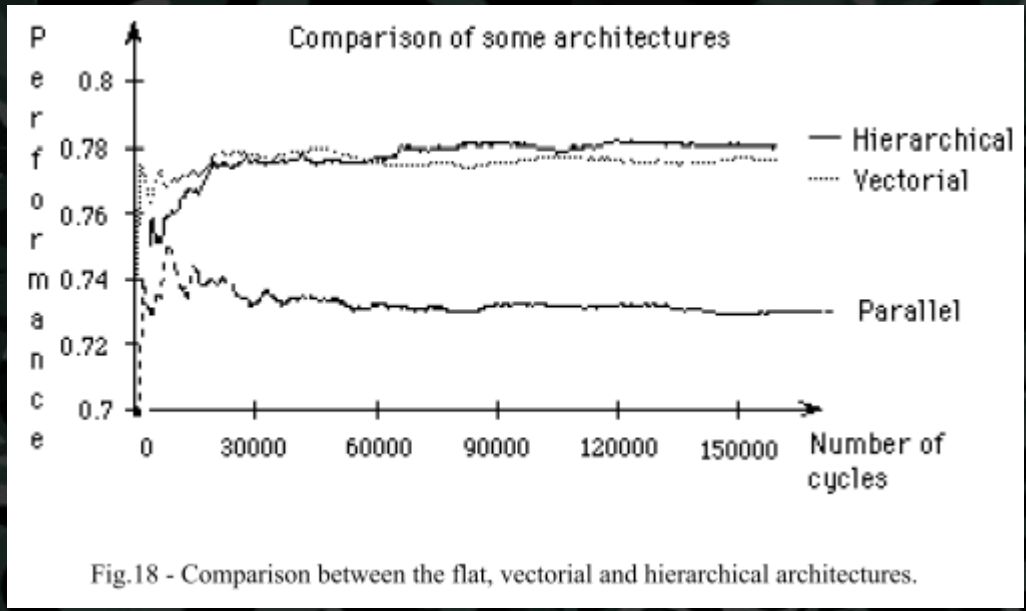

Fig.18 - Comparison between the flat, vectorial and hierarchical architectures.

Three separate tasks

The final test was to use rob2, which had three separate tasks to perform.

It had to follow the light, avoid a predator, and find food. These are often contradictory goals, so it has to be able to choose which one to do.

The LCS for each system must be implemented separately, then a higher level LCS will act as a switch to determine which one to follow.

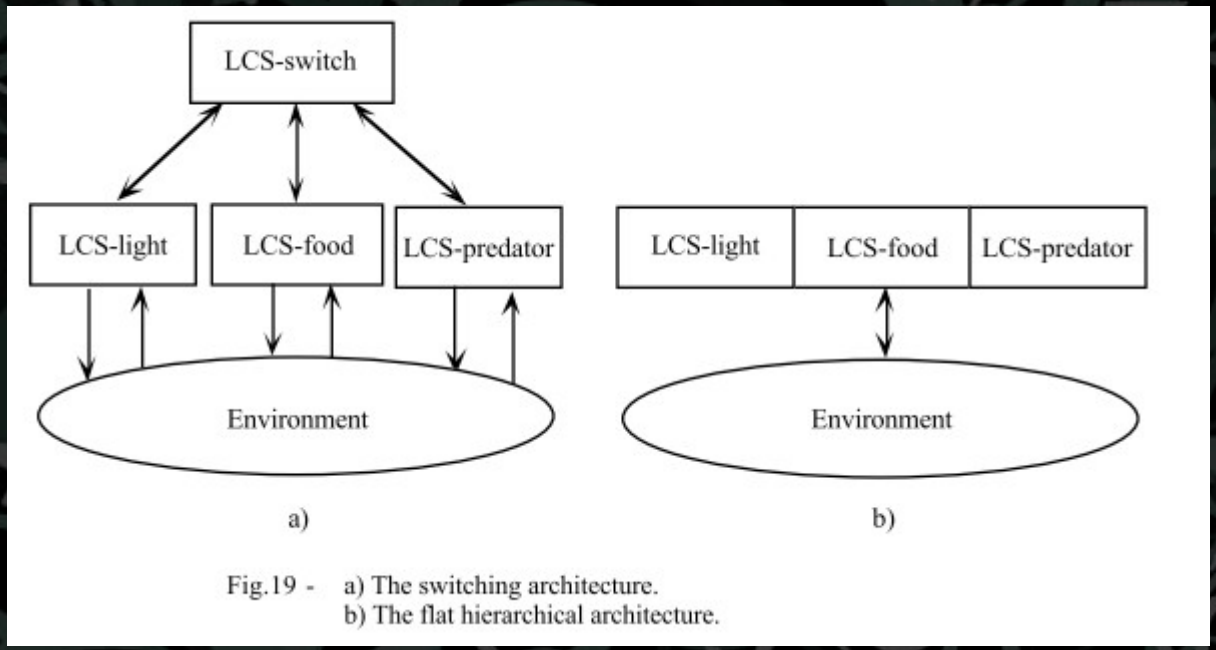

Fig.19 - a) The switching architecture.
b) The flat hierarchical architecture.

Clarifying the problem

The high level LCS receives 3-bit inputs, which tell it whether each sub LCS is proposing an action.

The high level LCS is supposed to coordinate which one is active: predator takes priority, then the other two are decided.

There are many ways to let the system learn, they used two; letting it learn contemporaneously, or to train the low level LCS then freeze them, and learn the high level LCS.

They will measure both flat performance, which is the performance of a subsystem while active, and global performance, which is the performance of the system as a whole.

Flat architecture

The flat architecture takes in all the inputs into one LCS, which determines the behavior. Escaping a predator is easy because there are more directions away from something than toward it in a 2d space. Finding food is the most difficult task

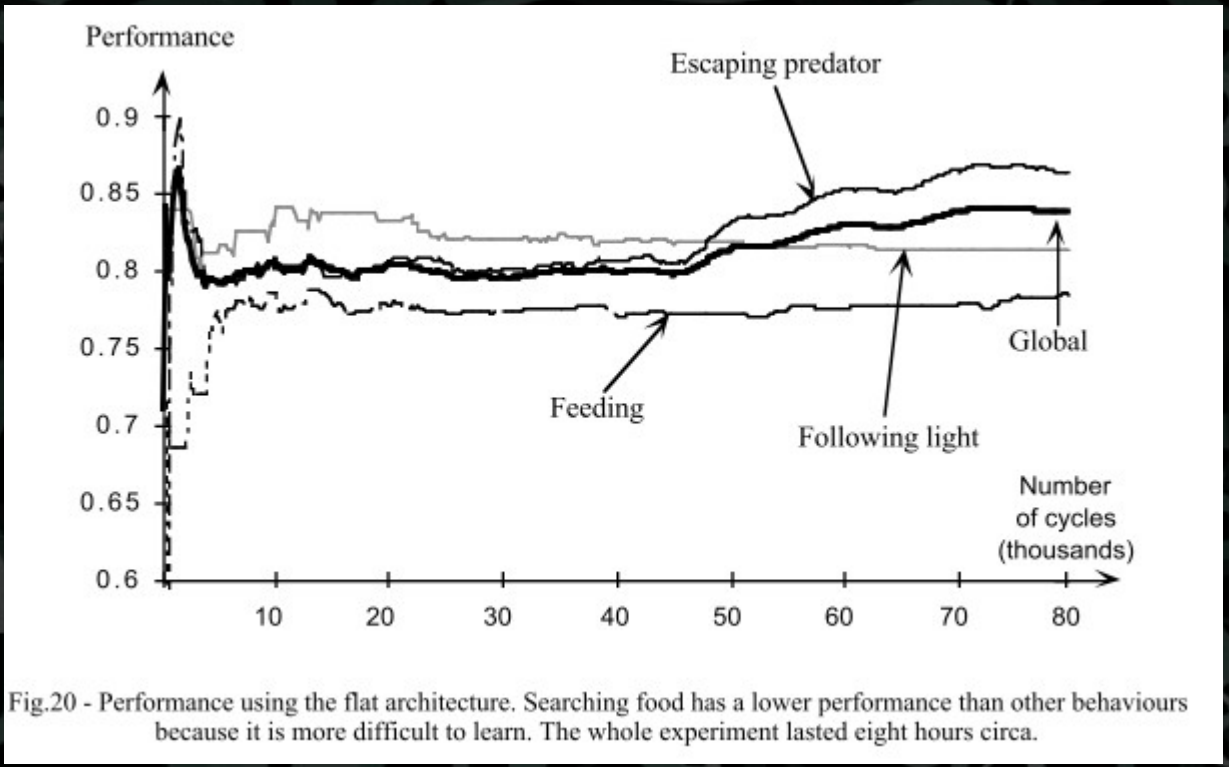

Fig.20 - Performance using the flat architecture. Searching food has a lower performance than other behaviours because it is more difficult to learn. The whole experiment lasted eight hours circa.

Concurrent Learning

Contemporaneously learning the system yielded individual performance similar to that of the flat learning system, and the overall performance is slightly lower, most likely because of noise in the reward function.

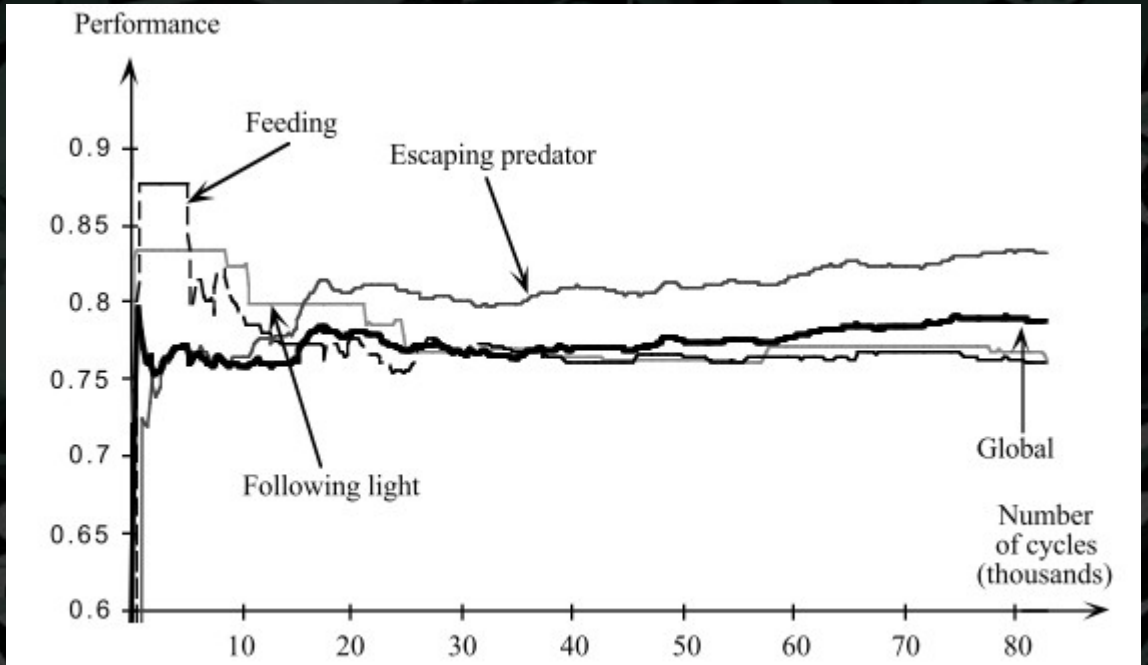

Fig.21 - Performance using the switch architecture. The whole experiment lasted eight hours circa.

Two Phase Reward Policy

The performance with the two-phase system is much better than the concurrent learning. It let the individual TCLs learn till they are good, then locking them while the switch learns. The results are better in performance and cycles to learn.

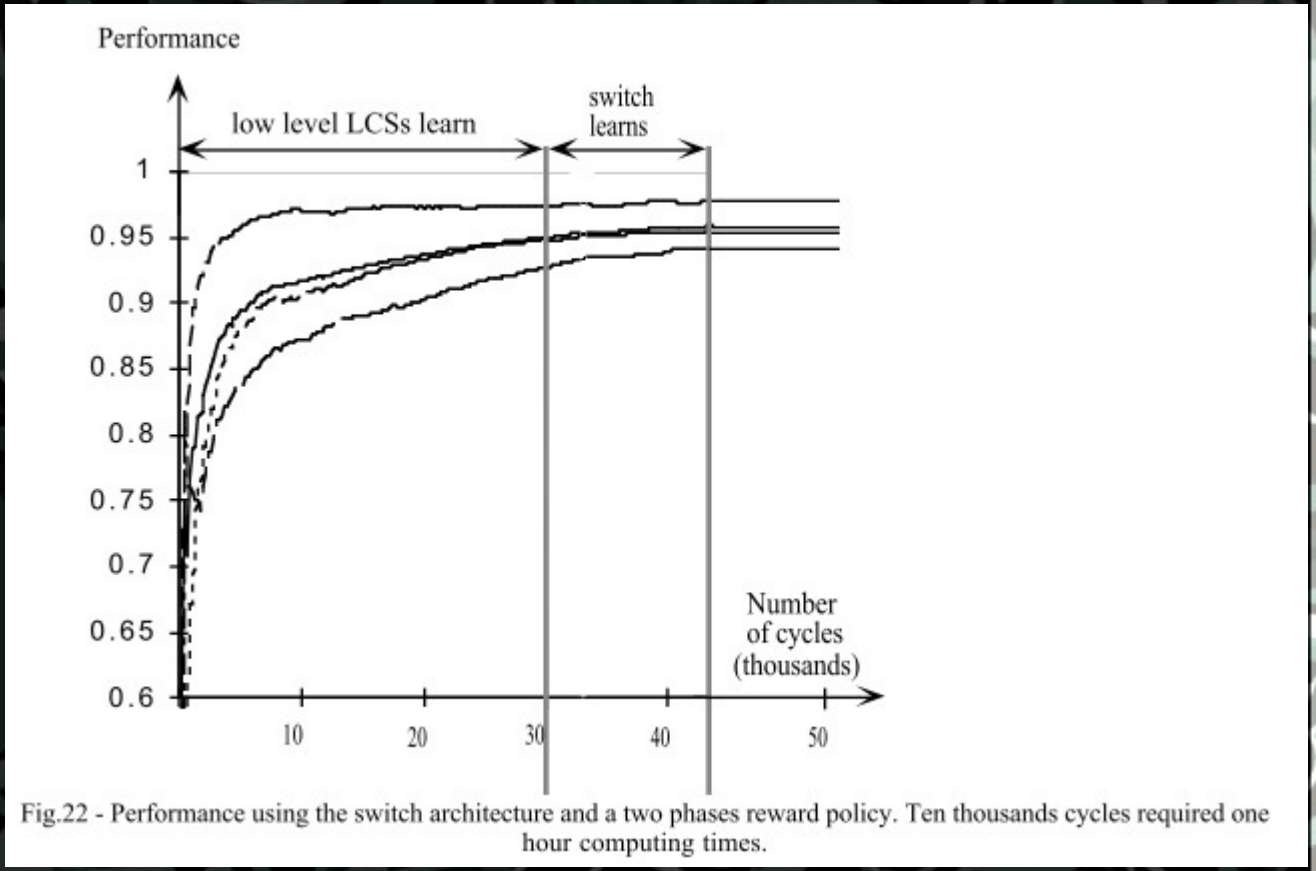

Fig.22 - Performance using the switch architecture and a two phases reward policy. Ten thousands cycles required one hour computing times.

Conclusion

This paper showed that a learning system can learn basic behaviors, and can learn how to combine them to create more complicated behavior. They showed that the method used to train the system makes a significant difference in how fast and how well it can learn its behavior.

The adaptability of such a system to more complex behaviors is possible in theory, and works much better than a rigid system. They did not implement their complete system fully, so its full potential has not been tested, but working as a subset of it it shows potential.

They did later research in applying this to practical applications, such as controlling a robotic arm, and it worked well, except they did not have a high enough data transfer rate between their hardware components, so dealing with moving objects was not possible. Besides this technical shortcoming the system worked remarkably well.