LEARNING CLASSIFIER SYSTEMS IN AUTONOMOUS AGENT CONTROL TASKS

Arkady Borisov* and Alexey Vasilyev**
*Decision Support Systems Group
Riga Technical University
1 Kalku Str., Riga LV-1658, Latvia
**Department of Computer Science
Transport and Telecommunication Institute
1 Lomonosov Str., Riga LV-1019, Latvia

This study suggests a classifier system using a partial model of environment for decision making. The more complete the model is, the better performance the system has. In the case when there is no model of environment, the system operates as a common classifier system using only local sensory information. The system has shown quasi-optimal results for quite complicated discrete environments in the task of autonomous agent control.

Keywords: adaptive behaviour, learning classifier systems

1. INTRODUCTION

Recently, strong support has been given to such direction of Artificial Intelligence as adaptive behaviour, implying the presence of an agent capable of being adapted to the environment in the process of its operation. Typical representatives of the adaptive behaviour are classifier systems [1] based on syntactically simple rules.

The adaptive agent has to examine the environment independently, having only some general information on its nature. For that, it has to sense the environment by sensors, build a presentation of the environment in the memory and execute some actions.

2. THE CONCEPT OF CLASSIFIER SYSTEM WHICH USES PARTIAL MODEL OF ENVIRONMENT

The main problem of learning classifier systems (LCS) is that they only use sensory information, while LCS for environment models do not give optimal results. Classifier systems without memory cannot find an optimal path in non-Markov environments, since their local perception in most cases does not give a univocal estimation of optimal policy π. Using the memory gives only minor improvements and strongly depends on the memory size [2].

As one of the possible solutions to the problem, let us examine a learning classifier system using a partial model of environment (LCSME). The environment model assumes the existence of graph G, describing environment states and using transition probabilities $p_{ij} > 0$ from state s_i into adjacent state s_j. The partial model means that the environment can be unexplored for some states s_i, *i.e.* transition probabilities in these states are the same, $p_{ij} = \text{const}, \forall j$.

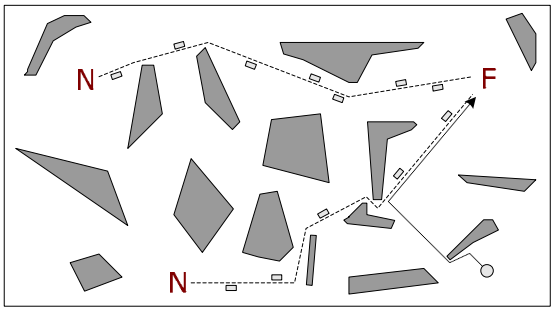

Fig. 1. Classifier system navigation using partial environment model created by ant colony

The partial environment model is formed before classifier systems LCSME begin operating which are based, for instance, on artificial ant colonies. In this case a randomly appearing agent searches for the nearest path where artificial ants walk. As a rule, ants walk along the shortest path from nest (**N**) to food (**F**). When the agent has found this path, it goes along it towards food (**F**) (see Fig. 1).

3. CLASSIFIER ACTIVATION

LCSME at each step t receives information from the environment on the current state s_t and transition probabilities from the current state to adjacent states $p(s_t, s_{t+1}), \forall s_{t+1} \in N_t$.

Parameter h' that is responsible for a classifier's fitness increases depending on the value of transition with the same action, which increases the probability of classifier activation.

$$p_s = \frac{h + \gamma \cdot p \cdot h}{\sum_{j=1}^{|M|} \left(h^j + \gamma \cdot p^j \cdot h^j \right)}, \quad a_k(p) = a_k(\varsigma),$$

where $\gamma > 0$ is the factor of intensification/reduction in the influence of environment model on the classifier activation probability. When $\gamma \to 0$, the environment model is not used.

p – transition probability in the environment at execution of action a_k.

For example, if an agent is in state s_t and has the ensuing transition probabilities with regard to current $N_t = \{p_{west} = 0.1; \tau_{north} = 0.5; \tau_{east} = 0; \tau_{south} = 0.8\}$, then for the classifier <condition : action : fitness> $\varsigma = \langle \Upsilon = \text{"WE##W\#EE"}; a = \text{"north"}; h = 0.4 \rangle$, fitness value $h' = 0.4 + 0.9 \cdot 0.5 \cdot 0.4 = 0.58$, $\gamma = 0.9$.

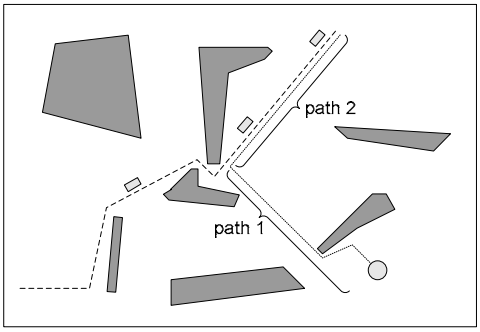

Fig. 2. Agent's navigation using partial environment model

In Fig. 2 two paths are shown. On the first path, activation of classifiers only depends on their strengths, while on the second path it depends on classifier strengths and on transition probabilities in the environment model.

3.1. Modification of existing classifier systems

To modify existing classifier systems one must change the procedure of classifiers' activation so that it depends both on the current classifier fitness and on the transition probability in the environment model.

For instance, for the ACS (Anticipatory Classifier System) [3] activation of classifiers depends on the quality r of classifier and prediction q of reward, *i.e.* classifier fitness $h = r \cdot q$. Classifier fitness which uses the environment model is computed by formula $h' = r \cdot q + \gamma \cdot p \cdot r \cdot q$. This kind of classifier system will be called ACSME (Anticipatory Classifier System which uses Model of Environment).

4. HEURISTIC ANALYSIS

As estimation of the efficiency of proposed learning paradigms, the so-called Animat problem (Animat=Animal+Robot) is used [5]. The essence of this problem is searching for immovable objects (*e.g.* food) in a maze. An agent's life consists of several cycles: it is placed in a randomly chosen spot, after which the agent has to find the sought-for object within the least possible number of steps.

For LCSME to operate, there must be a graph of the environment's model. The construction of the graph is based on the artificial ant colony [4].

4.1. Partial environment model construction by using ant colony optimization

An artificial ant colony generates a partial model of environment. Although ants use the environment model for navigation, the model is not fully available to them.

Each agent (ant) is characterized by the local state of environment described by eight-dimensional vectors. The agent has information about the amount of pheromones located on adjacent states N_i. The environment can have a large number of agents. In it there can be several nests (**N**) and food sources (**F**). The agents come from nest **N** in order to find food **F** and they leave pheromones on their way.

There are at least two types of solution to the navigation task in mazes if we use the artificial ant colony model.

- *The first type* of navigation implies that pheromones left by the agents are located in the states (similar to the situation with real ants). In this case all arcs entering state s_i nodes have the same value equal to that of pheromone located in this state s_i.

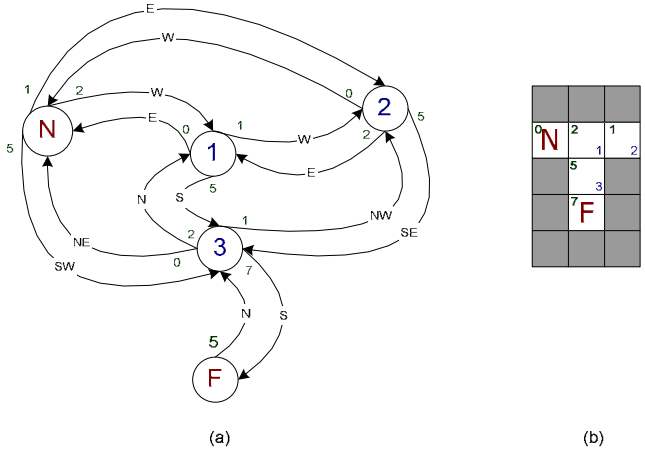

Fig. 3. Example of environment where pheromones are stored in the states

- *The second type* of navigation is an environment in which pheromones are located on the transitions from state s_i to state s_j, but not in the states. This type of coding gives better results. It will be used in further experiments.

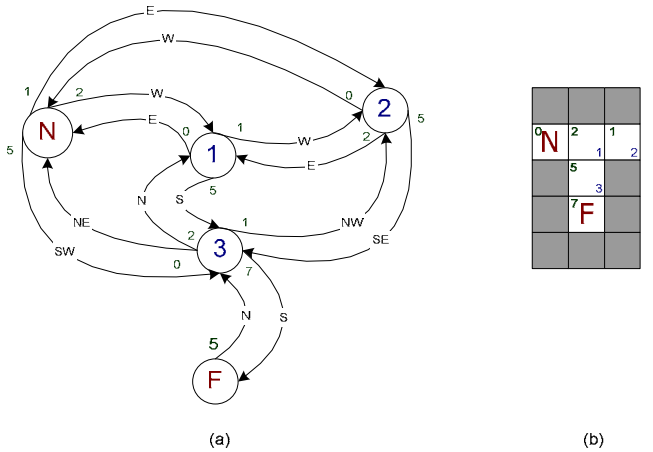

Fig. 4. Example of environment where pheromones are stored in the transitions from one state to another

Initialization of transition values by values equal to τ_0 greatly influences an agent's learning, since in the case of unexplored environment it allows agents to make decisions with equal probabilities, *i.e.* independently observe the environment. If the environment is initialized by random values, in most cases the agents will not be able to find food.

Pheromone values can be updated in two ways:

- *Single-pass (accumulation)*. In this case each agent remembers a certain number of the last states it is visited. When the food is found, the agent distributes pheromone values *ph* among the previously visited states. The agent has a memory.
- *Two-pass*. An agent leaves two types of pheromones: the *first* type is left when the agent does not carry food and the *second* one when it does. In order to find the way to the nest, the agent leaves the first pheromone ph_1, but searches for food by using the second pheromone ph_2. After the food is found, it lays ph_2, and searches for the nest by using ph_1. In this case the agent does not have a memory.

To prevent recirculation in the case when the agent's departure point is not an impasse, we will use a *tabu list* [6]. The tabu list for each state is the state from which agent *k* has just come.

$$tabu_k = \{s_{t-1}^k\}.$$

Islands of colonies are used in the case of a complex model, *i.e.* when we have a problem of high dimensionality. The colonies work on the same problem independently and synchronize the best solutions among themselves after an interval $syn_{islands}$ by using the following formula:

$$p_{ij}^{(n)} = \max\left(p_{ij}^{(m)}\right),\ m = 1,2,...,n_{islands}.$$

For each probability p_{ij} of colony *n* the maximum of transition probability is taken. This approach results in the distribution of the best solution information among the colonies if at least one colony has found such a solution.

The solution of the Animat task using an ant colony optimization (ACO) differs from that by means of LCS in that for each trial a random initial position is chosen in a maze. In the case of ACO the initial position (nest) is static.

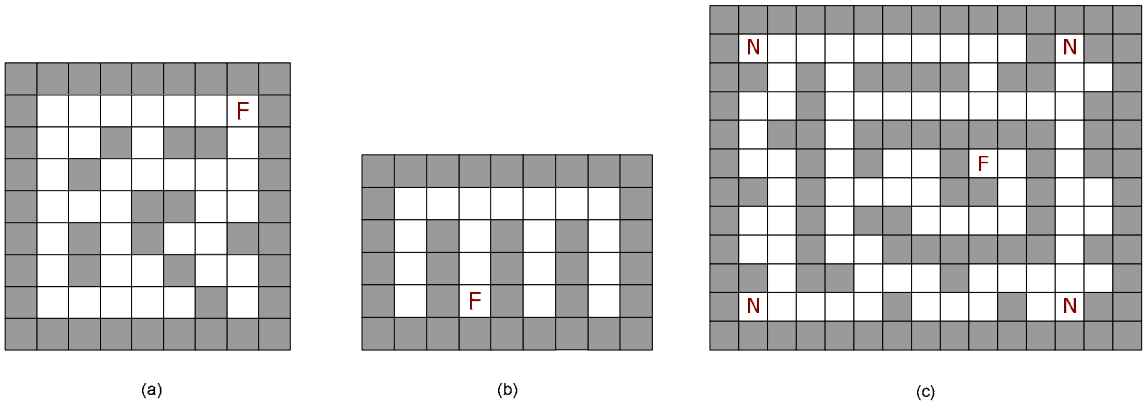

Fig. 5. Environment Maze5 (a), Maze10 (b), and Lab1 (c) with four nests (N) and one food (F)

The performance of the artificial ant colony model will be examined in testing environment Lab1 (Fig. 5.c) using the following parameters: the maximum number of transitions saved in an agent's memory $\max\left(\left|M^k\right|\right) = 50$, the pheromone decreasing factor at each iteration $\rho = 0.008$; the amount of pheromones left by each agent, $ph = 10$. Island synchronization occurs after the time interval $syn_{islands} = 2000$. The tabu search is applied. The graph is averaged over ten experiments.

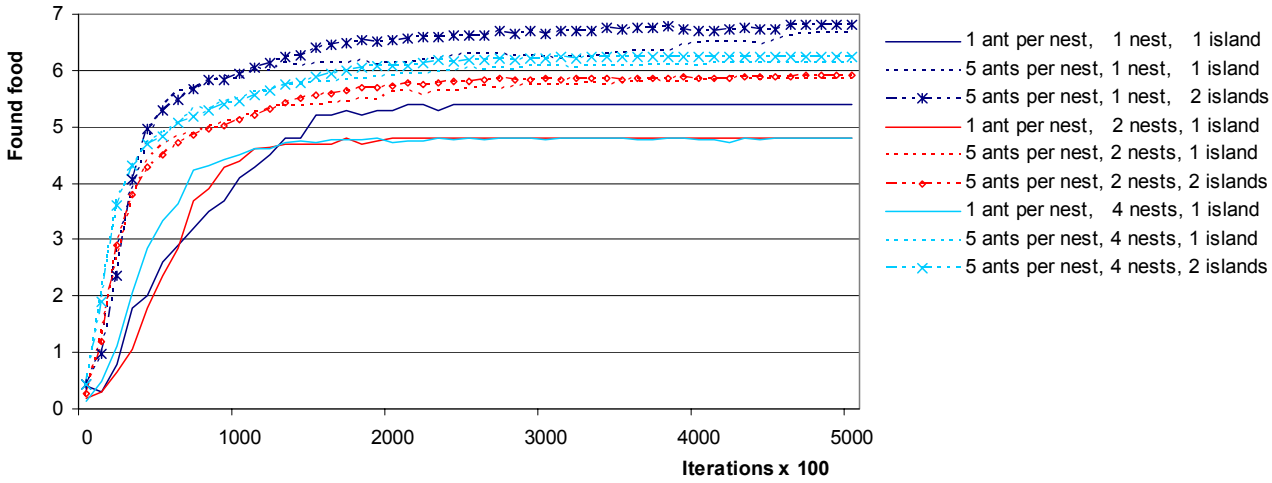

Fig. 6. Ant colony optimization performance in Lab1 environment

The artificial ant colony model shows the best performance when there is an increase in the number of simultaneously working agents and in that of islands (Fig. 6). For instance, in a single-nest environment with five agents and two independent colonies (islands) each agent at the 5000-th iteration finds food 6.82 times per 100 steps on the average.

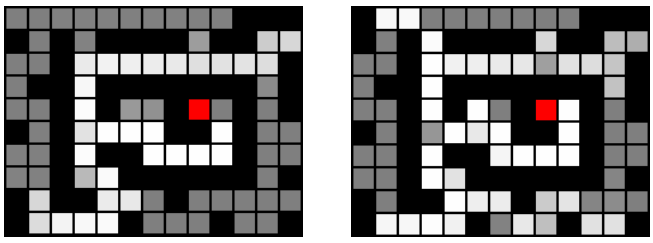

Fig. 7. Example of pheromons distribution in Lab1 environment with two nests (on the left) and four nests (on the right)

An example of pheromone distribution for environments with two and four nests is shown in Fig. 7. The lighter path shows the higher pheromone concentration.

Unlike the majority of the known reinforcement learning algorithms, the requirements imposed by ACO on the agent memory are minimal, since the agents do not possess graphs of the model.

4.2. Animat task solving

To solve the Animat task we will use classifier system ACSME and a partial graph of the environment model formed by the ant colony model. The parameters used by ACSME in experiments are default ones [7]. All the graphs are averaged over ten experiments.

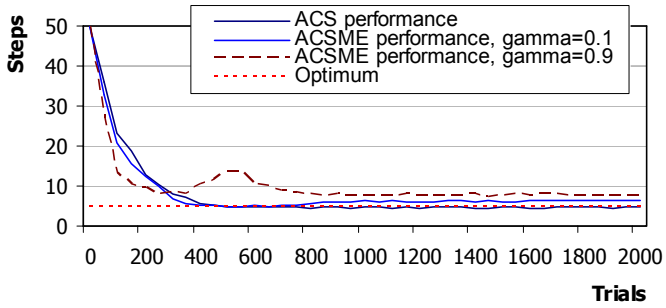

Fig. 8. ACSME performance in Markov environment Maze5

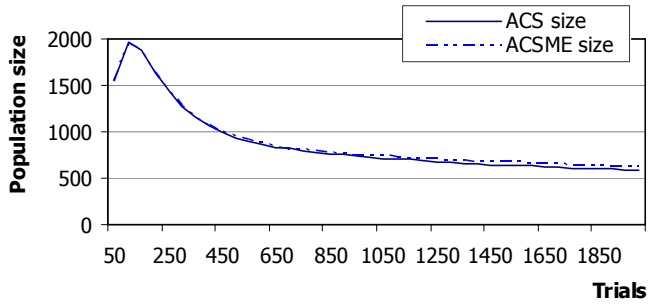

Fig. 9. Variation of a classifier's population size in Maze5

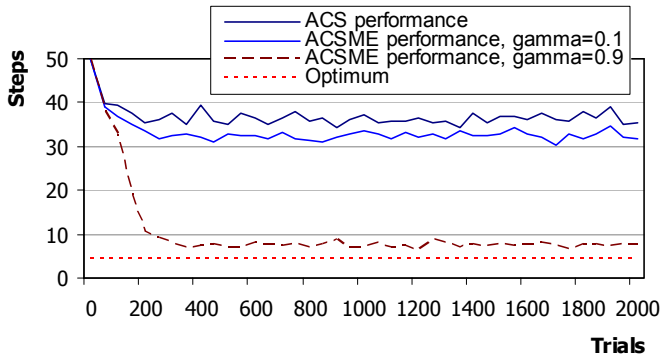

Fig. 10. ACSME performance in non-Markov environment Maze10

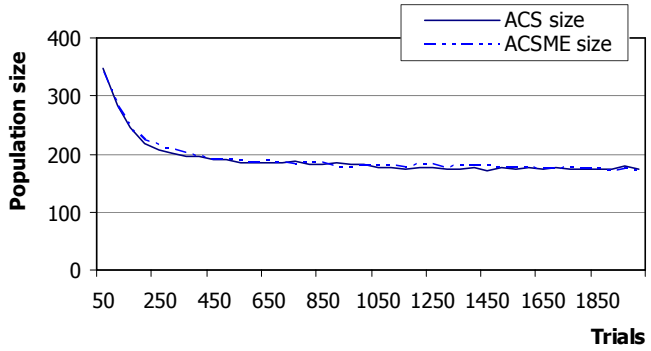

Fig. 11. Variation of a classifier's population size in Maze10

For sufficiently simple environments, such as Maze5 and Maze10 (see Fig. 5.a and Fig. 5.b), classifier system ACSME shows a quasi-optimal performance (see Fig. 8 and Fig. 10). However, in contrast to ACS, the ACSME system can effectively operate in non-Markov environment Maze10.

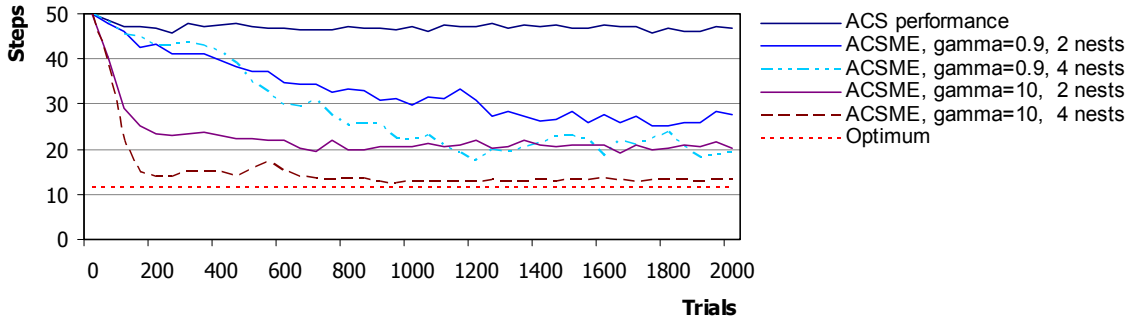

Fig. 12. ACSME performance in non-Markov environment Lab1

Analysis of ACS operation in a more complex non-Markov environment Lab1 shows that it is unable to learn efficiently. The ACSME system, on the contrary, shows good results (Fig. 12). When there is a partial graph of environment, created by an ant colony with four nests (5 agents per nest) with coefficient γ=10, classifier system ACSME at the 200-th iteration is able to find the goal object (**F**) on the average within 13 steps (optimum=11.3 steps).

5. CONCLUSIONS

In this work, a new classifier system (LCSME) is presented which uses for decision-making both the sensor model and the environment model, and the latter can be partial. In unexplored states, *i.e.* with $p_{ij} = \text{const}, \forall j$, LCSME makes decisions according to the local sensory information. In this case, the system is functioning as a simple classifier system which uses only local perception information.

The LCSME performance is tested for the agent navigation task (Animat) in quite complex non-Markov environments and the results are considered quasi-optimal. Classifier systems which use only sensory information were unable to learn under these conditions.

REFERENCES

1. Holland, J. H. Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press, 1975.
2. Lanzi, P.L., Wilson, S.W. Optimal classifier system performance in non-Markov environments. Technical Report 99.36, Dipartimento di Elettronicae Informazione - Politecnico di Milano, 1999.
3. Butz, V. M. An Implementation of the Anticipatory Classifier System ACS2 in C++. Technical Report 2001026 at the Illinois Genetic Algorithms Laboratory, 2001.
4. Corne, D., Dorigo, M., Glover, F. (ed.). New Ideas in Optimization. McGraw-Hill, London, pages 450, 1999, ISBN: 0077095065.
5. Wilson, S.W. The Animat Path to AI. In From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour, pp.15 - 21. MIT Press/Bradford Books, 1991.
6. Maniezzo, V. & Carbonaro, A. Ant Colony Optimization: an Overview. Proceedings of MIC'99, III Metaheuristics International Conference, Brazil, 1999.
7. Stolzmann, W. An introduction to anticipatory classifier systems. P.L. Lanzi, W. Stolzmann, and S.W. Wilson (Eds.): LCS'99, LNAI 1813, pp. 175-194, 2000.