In: U. Rueckert, J. Sitte and U. Witkowski, editors, Autonomous Minirobots for Research and Edutainment, Proc. of AmiRE 2001, HNI-Verlagsschriftreihe.

W-Learning for Behavior Coordination on a Real Khepera

Dimitrij Surmeli and Horst-Michael Gross

Dept. of Neuroinformatics, Ilmenau Technical University,
PO Box 100565, D-98684 Ilmenau, Germany

E-mail: Dima.Surmeli@informatik.tu-ilmenau.de

Abstract. We utilize HUMPHRYS' W-Learning on a real robot Khepera to coordinate two behaviors in a maze: first, to avoid obstacles and second, to find a location marked by a certain color (e.g., where food can be found). We describe the experimental setup and compare results of the individual agents with one monolithic agent solving both tasks and the agents coordinated by W-Learning. We demonstrate the feasibility of W-Learning on a real robot.

1 Introduction

Reinforcement Learning [8] has had impressive success in a number of invesigations and applications. When using the variants employing value-approximation techniques, however, more is learned than just which the best action in each state is: for every action in each state, a value is obtained. While not strictly necessary to determine that best action, these may serve to coordinate a number of agents that share the same 'body'. Thus, the access of several agents to the effectors must be coordinated. This paradigm both shares some similarities and also differs from approaches in Distributed Artificial Intelligence, where the focus rests with either coordinating a multitude of complete individuals or subdividing a big, possibly partially observable task into a number of more manageable smaller subproblems, whose sequence must be determined. Further, there is a vast literature on multi agent systems in the field of symbolic Artificial Intelligence, which can neither be reviewed nor shortly summarized justifyably.

Here, the approach more resembles one that might be taken to model an entire creature with several needs and goals. The overall task for the creature might be termed 'survival'. We follow BROOKS in his behavior-based approach [1], which is also fairly well established. In our quest for an adaptive, intelligent coordination for multiple goals that goes beyond Brook's and similar approaches, we decided to investigate W-Learning [3] and its applicability on a real robot.

Of course, given a smart global reward function, that overall task may be solved by a monolithic learner at the cost of an immense state-action space and a great amount of time needed to learn. Scalability to complex tasks and acceptable adaptation times would present certain problems for such monolithic learner. The curse of dimensionality and an easier design of partial reinforcement functions are the driving forces for Multi-Agent systems.

2 Theoretical background

In this section, we provide a brief sketch of the algorithms employed to control the Khepera, along with the principal structure of the agents used.

2.1 W-Learning

In W-Learning [3], a collection of individual agents is recruited to each solve an independent task characterized by its own input space, its own reward signal and possibly, also its own action space. In its original formulation, W-Learning (WL) assumes that all these agents are fully trained Q-Learners (see section 2.2). It uses the Q-values acquired by these agents to determine just how badly they want to execute their own best action in a certain state.

Essentially, in each state an agent A_i submits its action a_i accompanied by a bid or weight W_i to a meta agent, which selects an agent A_k to execute its suggested action: $W_k(x) = \max_i W_i(x)$.

HUMPHRYS' discusses a number of variants as to how to determine these weights W_i. In the preferred approach, the W-value is determined from the difference between the reward expected from execution of the agent's suggested action, and the reward incurred from the action that was actually executed by another agent: $W \leftarrow D - F$, where D is the expected reward, and F is the experienced reward. A_i will transition into a new state b after execution of the possibly unknown action of another agent ruling the animat. Then, F is determined as usual in Q-learning as the value of that state.

$$W_i(x) \leftarrow Q_i(x,a_i) - (r_i + \gamma * \max_b Q_i(y,b)) \tag{1}$$

This variant, HUMPHRYS calls 'Minimize Worst Unhappiness' and uses this name interchangably with WL.

The effect may be described as follows: by not insisting on its own suggested action and relinquishing ownership of a state to another agent j, who executes an action that to agent i holds less reward, an agent i may suffer some slight loss, but possibly avoids catastrophic losses for j. For the animat at large, this is a preferable solution.

2.2 Q-Learning with a neural vector quantizer

Among reinforcement learning algorithms, Q-Learning [9, 10] is the one algorithm that has been investigated most extensively. Equation 2 is provided as a short explanation.

$$Q(x,a) \leftarrow (r_i + \gamma * \max_b Q_i(y,b)) \tag{2}$$

where the value Q in state x for action a is determined from the immediate reinforcement experienced r_i and the value of the best action b in the subsequent state y. Both immediate reinforcement and the following state are subject to probabilistic transitions resulting from action a in state x and therefore, represent samples from corresponding distributions.

Here, we employ Q-Learning in a straight forward manner. We set a number of actions available to each agent in a fixed number of states. These states are induced from the continuous input stream by means of a vector quantizer, such as a Neural Gas. We obtain the Q-values from the standard temporal difference error in a supervised layer subsequent to the output of the Neural Gas [5]. Details of that setup can be found in [2] and [7].

The Neural Gas is an optimal vector quantizer which realizes an unsupervised mapping similar to that of Kohonen's well known Selforganizing Feature Map (SOFM, [4]). The difference pertains to the internal structure of the net: in a Neural Gas, there are no fixed neighborhood relations whereas in SOFM, a grid is enforced. Subsequently, no mapping errors occur when the dimension of the neighborhood structure is smaller than that of the input. The weights of a Neural Gas move free in the input space. All weights will move towards the current input vector in nonlinear proportion to their ranked distance to it.

2.3 Control of an agent by Q-Learning

Put together, an individual agent works as follows:

1. obtain sensory input
2. present input to Neural Gas
3. adapt Neural Gas weights
   (a) calculate difference of all nodes to input vector in Euclidean Space
   (b) rank nodes accordingly
   (c) determine state as Best Matching node
   (d) move weights
4. determine state value and action to be executed from Q-values connected to Best Matching node
5. execute action
6. receive reward, calculate new state and its value
7. update Q-values of last state by Delta-rule

3 Experimental details for the agents

Our experiments are realized on a Khepera miniature robot linked via serial cable to a host computer, where all processing is executed. The robot is equipped with the standard 8 infrared sensors and features an additional omnidirectional color camera turret supplying visual information about the environment. The preprocessing of the sensory inputs and their assignments to the agents is discussed in the following sections.

The action repertoire of the robot for all experiments and all agents consisted in locomotion only. An action, expressed as a vector, was comprised of a value for speed and turning angle, respectively. *speed*$\in [-1,4,7]$; *angle*$\in [-60,0,60]^\circ$ Every combination was allowed and turning was realized as motion of both wheels in opposite directions.

3.1 Obstacle Avoider

One task is to avoid obstacles. The sensory information supplied towards this goal consists of the readings of the 8 infrared sensors in active mode, averaged over 10 measurements for one reading. The readings were normalized to $[0\ldots1]$. Figure 1 provides a sketch of the agent and the location of the infrared sensors on the Khepera when viewed from above.

The avoidance agent was punished for hitting an obstacle, but also for getting too close to it. Initially, only those events were punished. As that resulted in standing still in first tests, that action was disallowed by restricting the action space as noted above. Next, it found turning in place to be optimal, such that we decided to also punished it for turns and reward it for moving forward, depending on the speed. We did not perform a systematic, careful tuning of the rewards emitted for any agent, and used here the reinforcement function in eqn. 3

$$r(Obst) = \begin{cases} \text{-5.0}, & \text{if } \max_{IR} > 0.9 * MAX(IR) \\ \text{-3.0}, & \text{if } \max_{IR} > 0.8 * MAX(IR) \\ & speed * 2.50 - |(0.2 * angle)| \end{cases} \quad (3)$$

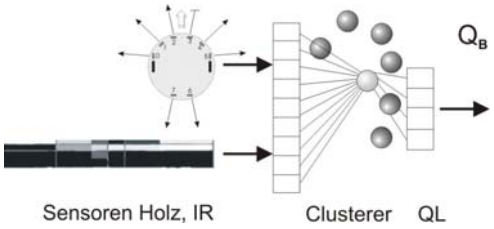

Figure 1. Schematic structure of the Obstacle Avoider

3.2 Food Seeker

The second task, finding food, was realized based on visual input exclusively, which in figure 2 is symbolized by a camera. The agent's structure is similar to the Obstacle Avoider, only the input vector is longer.

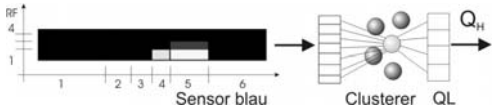

Figure 2. Schematic structure of the Food Seeker

The camera turret actually puts out an omnidirectional view of the environment (see fig. 3 left), produced by a camera looking straight up at a parabolic mirror. Mostly for reasons of human understanding, the image is transformed by a logarithmic polar transformation to obtain a more normal rectangular image (right top). To reduce the data represented to the vector quantizer, a grid of 6 by 3 areas was defined, which could be interpreted as receptive fields. For each of these fields, one average color was computed by weighting all pixels by a Gaussian centered over the middle of the field, with a variance corresponding to the width of the field (right bottom).

To reduce the influence of varying lighting conditions, these colors were then transformed into the physiological color space [6] and normalized to $[0\ldots1]$. Thus, it would be possible to discriminate between ground, walls and the patch purely by color.

The Food Seeker was rewarded for stepping onto the green patch. Therefore, its reward depended on the number of green pixels in the image. That number was determined by counting green pixels in the lowest parts of the image. Further, to encourage minmal trial lengths, it was punished for every step it took. Thus, it would rather avoid long straight lines unless they lead to the green patch.

$$r(Food) = -3.0 + (numberofgreenpixels) * 0.1 \tag{4}$$

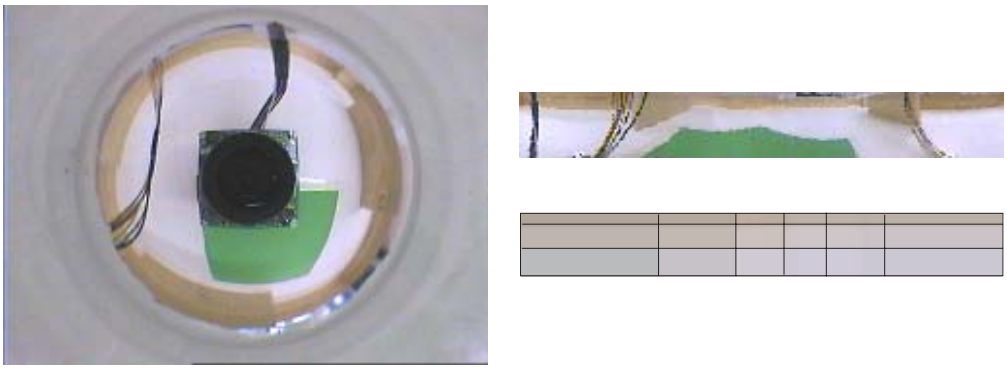

Figure 3. Processing of visual inputs. Source is an omnidirectional view (left), which is cleaned and log-polar transformed (right top). The data are reduced by averaging the colors weighted by 18 Gaussians arranged in a 6x3 grid (right bottom)

3.3 Monolith

The Monolith received as inputs the inputs of both the Obstacle Avoider and the Food Seeker. Therefore, it had to deal with a much more complex mapping. This requires more time to train, but it also gives it the chance to use the camera for obstacle avoidance and theoretically, the infrared inputs to find the green patch. Of course, the latter is fiction.

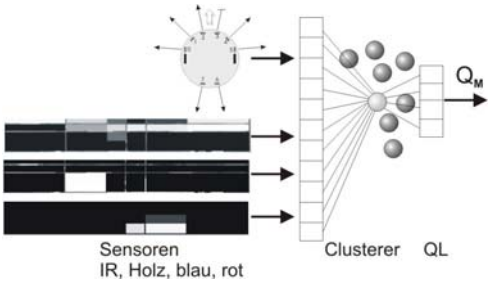

Figure 4. Schematic structure of the Monolith

Its reward was calculated as the sum of the individual rewards of the two individual agents:

$$r(Mono) = r(Obst) + r(Food) \tag{5}$$

3.4 W-Learner

The W-Learner was actually comprised of the very same agent networks used before for the Obstacle Avoider and the Food Seeker, complemented by a coordination by Negotiated W-Learning. In Negotiated W-Learning, no W-values must actually be learned as all agents share

the same action space. Thus, to determine their relative losses against each other, only the difference of their Q-values must be computed. As sketched in figure 5, each agent suggests an action and since all agents share the same action space, an agent can compute the loss it would incur if the action of another agent were executed from the difference of its own Q-values for its suggested action and those actions suggested by other agents. Both agents received their individual rewards as before. Performance of the entire agent was computed as sum, just like for the Monolith.

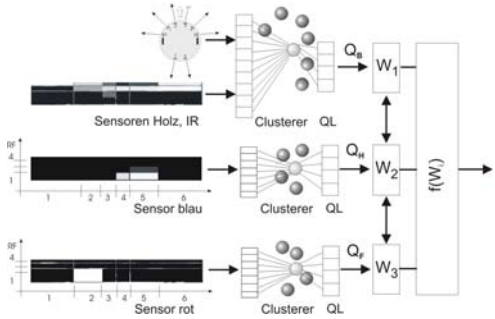

Figure 5. Schematic structure of the W-Learner

3.5 Learning

In correspondence to the approach taken by HUMPRYS, we first trained two Q-learners to acquire individually competence in their respective tasks. Both used standard Q-Learning ($\gamma = 0.8, \lambda = 0.0, \alpha = 0.5$) with the slight deviation that the learning rate α was held constant. Adaptation of the input clustering and the Q-values proceeded concurrently, and a Boltzman exploration was used. For the agent realizing obstacle avoidance, a Neural Gas of 50 nodes was used, thus 50 states were induced. The food agent was afforded 70 neurons. We gave the monolith 120 neurons to solve both tasks, just the sum of the individual agents and also, just the resources that the W-Learner had.

4 Results

We setup the maze with 5 points that we thought might be interesting starting points to compare the performance of the agents. We tested the performance of the four agents introduced above.

Every time, the Khepera was set into the scenario at the repective point in the orientation indicated by the white line in the dark circles in figure 6. All agents were allowed one trial at each point until collision, and the trial length was averaged for 5 experiments. The results are presented numerically for all agents in table 1 and discussed individually in the following paragraphs.

In these results, except for point 5, more (i.e. greater step lengths) is better. In point 5, where the food patch shall be reached, less is better. Of course, the obstacle avoider has no means to see the green patch, and its results are provided for completeness. Also, note that the food seeker is not interested in avoiding walls–whatever obstacle avoidance it achieves is the result of trying to minize steps. Also, it is not punished for turns and thus, has no preference for straights as the obstacle avoider.

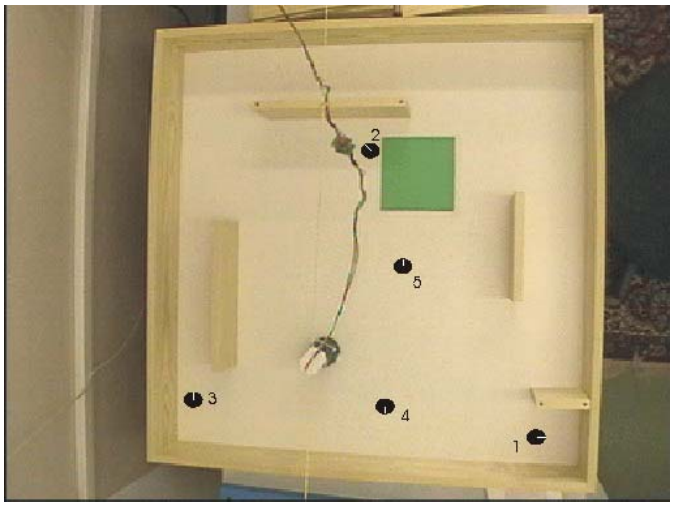

Figure 6. Khepera in its maze. The marks are the points from where performance was measured

Agent	Point 1	Point 2	Point 3	Point 4	Point 5
Obst Avoid	6	73.7	61.6	11.7	5
Food	37	3.7	9.3	107	17
Monolith	10.5	8.7	6	8.5	10.3
W-Learn	35	22.3	13.3	18.3	4.2

Table 1. Overview of all results. The numbers represent number of steps, either until collision or until the food patch is reached (relevant only for Point 5). No agent ever seized the opportunity for food from Point 2

4.1 Obstacle avoider

As the obstacle avoider 'sees' walls only from a distance of about 3cm away, it is mostly interested in driving long, straight lines and therefore, chooses the diagonals in the scenario. This also means it should try to avoid going into the dead end situation in the lower left corner (1) and rather chooses not to drive through the narrow passages (3). But it will go through them without collision. In fact, this presents a problem of partial oberservability: the avoider cannot distinguish a passage from a dead end, as its sensors do not reach far enough, as was observable in its behavior. In Point 4, when facing a wall straight up, again it is unable to comply with its task: it only sees the wall when it does not have room enough anymore to turn. From Point 2, it sucessfully avoids the wall and goes on to drive its beloved long straights, as well as from Point 5, which gives it superior performance to Food in finding the patch.

4.2 Food Seeker

The food seeker can actually see the green patch from a distance of about 15cm, which is about the width of the patch itself. Therefore, for the remaining bigger part of the scenario, it faces fairly unmarked territory except for the walls, which it recognizes from further away than the avoider. As it is punished for every step and thus, will try to minimize the number of steps it takes, it avoids the dead end situation (1), which causes it to take more steps. Also, it manages the head-on situation in Point 4 better, but cannot actually avoid collision from Point 3. In Point 2, it does not recognize the reward patch right behind it. As it cannot be explained satisfactorily,

this disappointing behavior is being investigated in ongoing experiments. From Point 5, in most cases it reaches the green patch and stays on it, using turns and the very slow motion backwards. In the other cases, it is confused by the wires of the serial link cable. It is the only agent to stay on the patch, but it also takes the longest to get there, as the avoider, the monolith and the W-Learner all profit from the avoiders preference for straight runs.

4.3 Monolith

Any comparison between the monolith and number of coordinated agents is bound to be unfair. Then again, this is in fact part of our point. The monolith would need more than the sum of resources of the individual agents, as it does not know about the meaning of the two sensor modalities and therefore, is forced to build a full state space with both IR and visual inputs. In turn, this would call for longer training times.

4.4 W-Learning

The W-Learner was actually able to draw on the advantages of its constituent agents, which actually supported each other sometimes. Surprisingly, no certain picture can be established over a number of runs starting from the same points regarding ownership of those starting points in the sense that one agent always wins that point, and so a map of ownership of the state space could not be drawn. We attribute this to rea life variation when setting in the robot and time- and temperature variance in the sensors. However, for all points, there is a clear focus: the avoider takes points 1 and 3, while the food seeker wins in 2, 4 and 5. However, W-Learnings success at point 5 is due to the avoiders preference for long straight lines, as it takes over from the food agent right after the first step. W-Learning was the only agent to actually escape the dead end in point 1. All in all, the avoider wins most of the states.

5 Summary

We demonstrated the feasability of using W-Learning in conjunction with Q-Learning in a real, albeit laboratory, environment and compared it to a monolithic Q-Learner. Careful tuning of the reinforcement assignments seems indicated, as sometimes, agents benefit each other, while at other times, they will interfere, instead of W-Learning just profiting from the individual competence of the agents. However, it still supercedes a monolithic learner.

6 Outlook

This work represents the start of an investigation into the coordination of several agents and their goals. After running more experiments for a clearer statistical picture, and scrutinizing in details the ambiguous results for point2, we plan to increase the number of agents to include mobile food sources the robot would have to hunt, and predators it should avoid. However, we do not plan to invest heavily into planning capabilities (trap prey or avoid dead end situations). To do so in a very flexible manner, we devised the setup consisting of the labyrinth as used in these experiments, an aluminum frame bridging over it, a video beamer and a digital camera.

The frame carries both the camera and the beamer that allow us to project and control an arbitrary number of dynamic and stationary objects as patches of different colors into the scenario.

The input from the overhead camera is used to control the movements of those objects. They may follow the robot or flee from it.

We plan to investigate the emergence of agents from different reward sources and the acquisition of the minimal necessary input space to obtain that reward. We will substitute the rather technical assignment of reinforcement by dynamic homeostatic states of a simulated creature, such that when entering the food patch, the agent will satiate after a sufficient amount of time and the urge to feed will cease.

References

[1] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

[2] Horst-Michael Gross, Volker Stephan, and Hans-Joachim Boehme. Sensory-based robot navigation using self-organizing networks and Q-Learning. In *Proc. WCNN'96, World Congress on Neural Networks, September 1996, San Diego, U.S.A.*, pages 94–99. Lawrence Erlbaum Associates, Inc., 1996.

[3] Mark Humphrys. *Action Selection methods using Reinforcement Learning*. PhD thesis, University of Cambridge, Computer Laboratory, 1997.

[4] Teuvo Kohonen, editor. *Self-Organizing Maps*. Springer Series in Information Sciences. Springer-Verlag, Berlin Heidelberg New York etc., 1997.

[5] Thomas Martinetz and Klaus Schulten. A "Neural Gas" network learns topologies. In Teuvo Kohonen, Kai Mkisara, Olli Simula, and Jari Kangas, editors, *Artificial Neural Networks*, pages 397–402. Elsevier, 1991.

[6] Torsten Pomierski. *Neurophysiologisch motivierte Architektur zur Erzeugung stabiler Farb- und Texturreprsentationen*. PhD thesis, Ilmenau Technical University, Dept. of Neuroinformatics, 1996.

[7] Dimitrij Surmeli and Horst-Michael Gross. Benchmarking reinforcement learning based on neural function approximators. In *Proceedings of the Forth International Conference on Cognitive and Neural Systems*, page 22, Boston, 2000. Boston University Press.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An introduction*. MIT Press: A Bradford Book, Cambridge, MA, USA, 1998.

[9] Christopher J.C.H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[10] Christopher J.C.H. Watkins and Peter Dayan. Q-learning technical note. *Machine Learning*, 8:279–292, 1992.