A History-Based Approach for Adaptive Robot Behavior in Dynamic Environments

François Michaud
Department of Electrical and Computer Engineering
Université de Sherbrooke
Sherbrooke (Québec Canada) J1K 2R1
tel: (819) 821-8000 ext. 2107
fax: (819) 821-7937
michaudf@gel.usherb.ca
http://www.gel.usherb.ca/michaudf

Maja J Matarić
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
tel: (213) 740-4520
fax: (213) 740-7285
mataric@cs.usc.edu
http://www-robotics.usc.edu/~agents

Abstract

The behavior-based approach has proven to be useful in making mobile agents work in dynamic and complex situations. Since the behaviors are responsible for managing the interactions between the agent and its environment, observing their use can be exploited to model these interactions. In our approach, the agent is initially given a set of behaviors to choose from, and the algorithm adapts the selection of behaviors according to the history of behavior use, represented in a tree structure. The approach is validated using a vision- and sonar-based Pioneer I robot in non-stationary conditions, in the context of a multi-robot foraging task. Results show the effectiveness of the approach in taking advantage of any regularities experienced in the world, leading to fast and adaptable specialization for the learning agent.

1 Introduction

Various kinds of autonomous agents, especially in mobile robotics, have been designed using the behavior-based paradigm (Brooks 1986). This approach has been praised for its robustness and simplicity of construction. In a typical behavior-based system, the constituent behaviors are designed parsimoniously, executed in parallel, and prioritized using some fixed or flexible arbitration mechanism (Matarić 1997*a*). A fixed and parsimonious behavior set, however, does not allow an agent to easily adapt to changes in the environment, thus resulting in diminished flexibility (Maes 1989). The ability to adapt to changing dynamics, however, is especially important for agents operating in unpredictable and non-stationary environments. One possible improvement would be to dynamically select behaviors by introducing a cognitive component capable of modeling the environment (Michaud, Lachiver & Le Dinh 1996). When behavior selection is changed according to external and internal states, knowing when to use a behavior becomes as important as the behavior itself.

However, deriving a model of a dynamic and unpredictable environment is a very challenging problem. Careful considerations must be made of what to model, what representation to use, and what effect the model has on the decision process of the agent. These considerations must be as independent as possible of the agent's operating environment, and of the agent's own limitations in perception, action and processing. Since performance of situated agents is based on their ability to cope with and exploit the dynamics of interaction with their environment (Agre 1988, Brooks 1991), it is appropriate to model these interactions using a representation consistent with the control mechanism used, in our case the behavior-based framework. Doing so also preserves two key strengths of the behavior-based approach, situatedness and emergence (Brooks 1991). Modeling the interaction dynamics with the environment can be especially useful in the case of dynamic multi-agent/robot environments (Goldberg & Matarić 1997) where topological and metric models do not directly apply. These dynamics are difficult for the designer to characterize and fully predict *a priori*, and are thus best captured by the agent itself.

The idea explored in this paper is that interactions between an agent and its environment can be efficiently captured through observation of behavior use over time. The agent is initially given a set of behaviors, whose subset is used to safely accomplish the assigned tasks. The goal of the algorithm is to learn to select alternative behaviors that are not normally exploited, to adapt the agent's behavior to the experienced dynamics. By knowing the purpose of each behavior, the agent can autonomously evaluate and model the type of interactions it is having with its environment. The representation of the "interaction model" is based on the agent's history of behavior use, i.e., on any regularities in the sequences of behavior use that reflect the interactions between the agent and the world (Michaud & Matarić 1997, Michaud & Matarić 1998). The agent derives this model in a reinforcement learning fashion using a performance criterion also derived from behavior use. The proposed algorithm focuses on continuous life-time adaptation rather than on learning a one-time, static policy. Our approach is validated with a vision- and sonar-based Pioneer I robot in the context of foraging in a multi-robot environment.

This paper is organized as follows. Section 2 describes the experimental setup and the mobile robots used in the experiments, showing how behaviors can be organized to derive an interaction model and to change the agent's way of behaving. Section 3 gives a detailed explanation of the learning algorithm. Section 4 presents the experimental results. Section 5 summarizes related work, and Section 6 concludes the paper.

2 Experimental setup and task description

Our experiments were performed on a Real World Interface Pioneer I mobile robot (shown on the right of Figure 1) equipped with seven sonars and a Fast Track Vision System. The robot is programmed using MARS (Multiple Agency Reactivity System), a language for programming multiple concurrent processes and behaviors (Brooks 1996). The experiments were conducted in an enclosed 12'×11' rectangular pen containing pink blocks and a home region, marked with a green cylinder. These colored objects were perceived using the robot's vision system, which suffered from variations in light intensity at different places in the pen and from different angles of approach. Other obstacles and walls were detected using the sonar system; the dark stripe on the wall of the pen is a strip of corrugated cardboard used to improve sonar reflection. IS Robotics R1 robots, equipped with infra-red and contact sensors, were also used in the experiments; they were perceived as obstacles by the Pioneer's sonars.

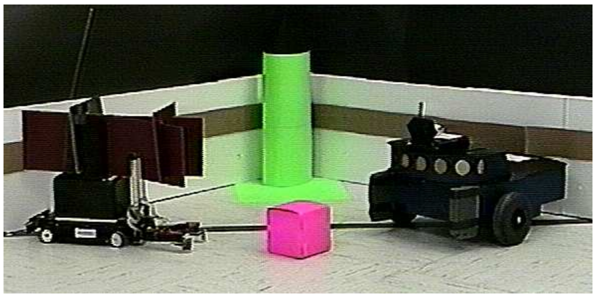


Figure 1: A Pioneer I robot (on the right) near a block, the home region and a R1 robot.

Foraging was done in two steps: searching for a block (Searching Task), and then bringing it to the home region (Homing Task). To do this, the Pioneer was given a set of behaviors and an initial controller that governed the use of some subset of those behaviors. The behaviors controlled the velocity and rotation of the robot based on sonar readings and visual inputs.

To analyze the interactions derived from observing behavior use, behaviors were classified according to their respective purposes, as follows. First, the robot needed behaviors to accomplish its tasks: one behavior for the Searching Task, called *Searching-block*, and two for the Homing Task, *Homing* and *Drop-block*. A behavior called *Velocity-control* was also used in both of these tasks to make the robot move. All these behaviors are referred to as "task-behaviors". Conditions for activating task-behaviors are pre-programmed based on the presence or absence of a block in front of the robot (using an internal variable indicating if the robot saw a block go underneath its field of view and is now being pushed) and the proximity of the robot to home.

Second, an *Avoidance* behavior was required for safe navigation in the environment. It was active unless the robot was near home and carrying a block; then it was disabled in order to let the robot approach the home region. This type of behavior, used for handling harmful situations and interference while accomplishing a task, is referred to as a "maintenance-behavior". The designer determines the conditions in which maintenance-behaviors should be used but cannot indicate when they will occur during the task, as that is tied to the dynamics of the interaction between the robot and its environment.

Third, the robot can also use behaviors that introduce variety into its action repertoire to change the way it accomplishes its tasks. We call these *alternative-behaviors.* In contrast to the other types of behaviors, no *a priori* conditions are given to activate alternative-behaviors: the objective is to let the robot learn when to activate these behaviors according to past experiences. In our foraging task, the Pioneer could decide that it was better to activate *Follow-side*, *Rest* or *Turn-randomly* for some pre-set periods of time when experiencing interference in accomplishing its task.

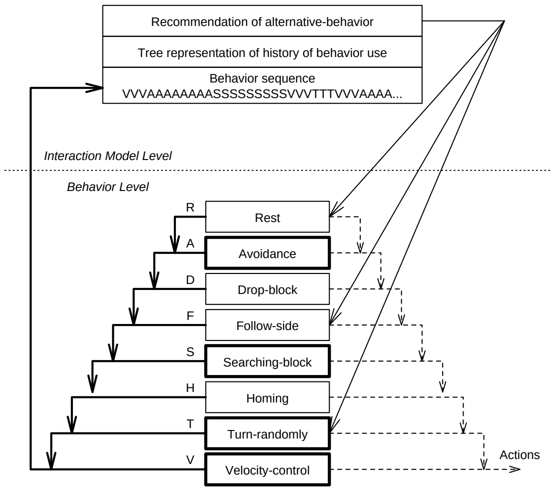
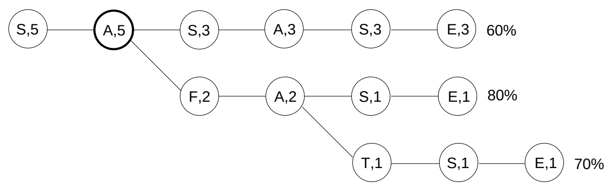


Figure 2: Overall organization of the behaviors and the model learned from observing behavior use. Activated behaviors for the Searching Task, with *Turn-randomly* as a chosen alternative-behavior, are depicted in bold.

The overall organization of the behaviors is represented in Figure 2, following the Subsumption Architecture (Brooks 1986) with the difference that the activated behaviors, i.e., those allowed to issue outputs, change dynamically. Behaviors that are not activated cannot participate in the control of the robot. The actual use of a behavior depends on the rules it contains and on commands issued by other behaviors subsuming it. These behaviors were not designed to be mutually exclusive: functionality can emerge by activating many behaviors at the same time (e.g., going home by following a wall). Also note that behavior classification is independent of the subsuming organization used in our experiments.




```
((S 5) (A 5)
 ((S 3) (A 3) (S 3) (E 3 60))
 ((F 2) (A 2)
  ((S 1) (E 1 80))
  ((T 1) (S 1) (E 1 70))))
```

Figure 3: Tree representation of the history of behavior use, followed by its list notation. For clarity, junctions are indented so the sub-paths are aligned.

3 History-based learning for adaptive behavior selection

Figure 2 also shows how behavior observation is used to learn the "interaction model". Using the subsumption mechanism, each behavior is assigned a symbol (e.g., F for Follow-side), sent through a wire whenever that behavior is executed. This generates a symbol sequence that represents the behavior use at each processing cycle, and is used to construct a tree representation of the agent's interactions with its environment and to recommend alternative-behaviors. Separate learning trees are used for each task; determining which one to use is done based on the activated task-behavior. Overall, the learning algorithm is composed of four parts, described next.

3.1 Tree representation mechanism

The algorithm uses a tree structure to store the history of behavior use. Figure 3 shows a typical tree in which nodes store the type of behaviors used for controlling the agent while accomplishing its task, and n, the number of times a transition between the node itself and its successor has been made (as observed from behavior use). Because there can be more than one task-behavior associated with each task, the node type for task-behaviors corresponds to the name of the predominant behavior for that task: a type S node represents the use of Searching Task behaviors, and a type H node is created when Homing Task behaviors are exploited consecutively. For synchronization reasons, the root node of a tree is always a task-behavior node, whatever is observed at the start of the task. For the maintenance-behavior and the alternative-behaviors, the letter associated with the name of the behavior (see Figure 2) is used as the node type.

Initially, the tree for a particular task is empty and is incrementally constructed as the agent goes about its task. Leaf nodes are characterized by the letter E (for end-node) and store the total *performance* of the particular tree path. Whenever a path is completely reused, i.e., when the same sequence of behaviors is executed, the average of the stored and current performances for the last 10 trials (this number was determined empirically from various experiments) is computed and stored in E. The history of behavior use and the resulting performance values provide the ability to learn to trigger alternative-behaviors according to the dynamics observed of the agent in the environment.

3.2 Metric for self-evaluation of performance

Performance can be evaluated using a variety of factors. Given an agent with limited capabilities operating in a dynamic environment, optimality is difficult to define and may also be non-stationary. As a possible solution to this problem, we explored time of behavior use as the key evaluation metric. However, other metrics could be used in different domains and for different tasks. Our objective was to see how far we could take the idea of self-evaluation and learning by observing behavior use. Our evaluation function, expressed by Relation 1 below, is characterized by the amount of time required to accomplish the task and the interference experienced during that process:

$$eval(t) = \max\left(0, \frac{(t_{tb} - t_{mb} + t_c)}{t_{tb}} - \max\left(0, \frac{(t - TT)}{TT}\right)\right) \tag{1}$$

where tb refers to task-behaviors, mb to maintenance-behaviors, and t to the current time-step (corresponding to one cycle of evaluation of all currently active behaviors). TT represents the average total number of time-steps taken to accomplish the task over the last 10 trials, updated upon completion of a task. Again, the number 10 was determined empirically by analyzing typical results; it establishes a compromise between adaptability to changes in the environment and tolerance for poor performance resulting from necessary exploration of different options. TT captures changes in the dynamics of the environment that affect the time to complete the task. t_c is a correction factor that increases when $eval(t)$ decreases below 0, to boost $eval(t)$ as soon as the agent uses its task-behaviors. The first term of Relation 1 reflects the amount of control the agent has over the task compared to the effect of other events that maintenance-behaviors must handle. The second term penalizes the agent's performance if it takes more time to accomplish the task than it has in the past.

3.3 Recommendation criteria

Using the tree and the evaluation function, the algorithm has different options: not to make any changes in its active behavior set (this option is referred as the *Observe* option), or to recommend the activation of an alternative-behavior. The sequence of nodes in a tree path characterizes the interactions experienced by the agent in the world, and situates the agent relative to those interactions. Different selection criteria can be used by comparing the performance at the current position in the tree (i.e, $eval(t)$) with the expected performance. The expected performance of a given node is the sum of the stored end-node performances in its sub-paths, multiplied by the frequency of use of the sub-paths relative to the current position in the tree. Formally, the expected performance for sub-path i starting from a given position in the tree is given by Relation 2:

$$E_i(eval) = \sum_j eval(T)_j \cdot \frac{n_j}{n_i} \tag{2}$$

where j refers to the end-nodes accessible from sub-path i, $eval(T)_j$ is the performance memorized in these end-nodes, and n is the number of times the node j of sub-path i was visited. This gives the expected performance for each sub-path in a tree, i.e., for each possible option at the current position in the tree. This formula can also be used to determine the expected performance at this position, $E(eval)$.

A simple example can be given by using the bold node in Figure 3 as the current node. The expected performance of the sub-path starting with S node is $60 \cdot 3/3 = 60\%$, while the expected performance of the F sub-path is $(80 \cdot 1/2) + (70 \cdot 1/2) = 75\%$. The total expected performance $E(eval)$ for this node is $(60 \cdot 3/5) + (75 \cdot 2/5) = 66\%$. Note that longer paths do not necessarily mean lower performance; the evaluation function is not influenced by the number of different nodes in the path but by the time spent in behavior use.

Using the expected performances at the current position in the tree and those at each of its sub-paths, the choice made by the algorithm is based on the following criteria:

- If $(eval(t) \geq E(eval))$, then take the option with the maximum expected performance $E_i(eval)$. This criterion is referred as the TM criterion, for *Try Maximum*.
- If $(eval(t) < E(eval))$, then two possible situations exist:
  - If there are untried options at this junction, then select an untried option. This selection criterion is referred to as UO, for *Untried Option*.
  - If there are no untried options at this junction, choose the one with overall best performance (i.e., over all the trials up to that point). The overall best option is obtained from a *Global Options Table* (GOT), an array of past performances indexed by the name of the option. The GOT is updated at the end of each trial by averaging the resulting performance with the ones stored, for all the options used for that trial.

The evaluation of the recommendation criteria takes place only at a maintenance-behavior node (i.e., node created by the observation of a maintenance-behavior) because that is when interference occurs in the accomplishment of the agent's tasks, resulting in decreased $eval(t)$. When one of these nodes is reached, the recommendation can be directly set by UO or GOT if $eval(t)$ is already lower than $E(eval)$, or by TM if $eval(t)$ is greater or equal to $E(eval)$. If this condition changes at the given position in the tree, UO or GOT can override the selection made using TM. Note also that when a new path is constructed, its expected performance is 0 and the default option is to behave without using any alternative-behaviors. This allows the agent to explore what can be accomplished by using task- and maintenance-behaviors only.

3.4 Path deletion mechanism

Since this algorithm is used in noisy and non-stationary conditions, deleting paths is necessary to keep the interaction model that the tree represents up to date. More specifically, because the expected performance is a weighted average based on the frequency of occurrences of the sub-paths, the most tried option at a given maintenance-node will have a greater influence on $E(eval)$. This decreases the variation of $E(eval)$ as an option is tried frequently, making it harder to modify it and to change strategies if the option becomes less useful. Forgetting paths may lead to exploring some of the choices again and discovering changes in the world. Node deletion also serves to regulate memory use. To enable the agent to respond to recent changes in the environment, we opted for deleting the oldest path in the tree when more than 25 different paths were stored. Other criteria can be easily substituted for different applications.

4 Experiments

Although the goal of the proposed algorithm is to learn in dynamic environments, it would have been inappropriate to start directly with a multi-robot domain experiment where non-stationary conditions make the analysis and the validation of the properties of the algorithm extremely difficult. To gradually address this problem, we first used static environment conditions with increasing complexity to verify that the algorithm could learn stationary conditions from its interactions with the environment. Next, experiments with multiple robots were performed. The same behavior repertoire was used in both sets of experiments, without optimization or retraining of the vision system parameters for color detection. The objective was to create situations where the designer could not know or adjust the behaviors according to *a priori* knowledge of the environment. All the computations were done on board, with a limited amount of memory (the entire robot program used 66 Kbytes of memory, with 27 Kbytes used by the learning algorithm, leaving 43 Kbytes for storing the trees; a tree of 200 nodes takes approximately 7 Kbytes).

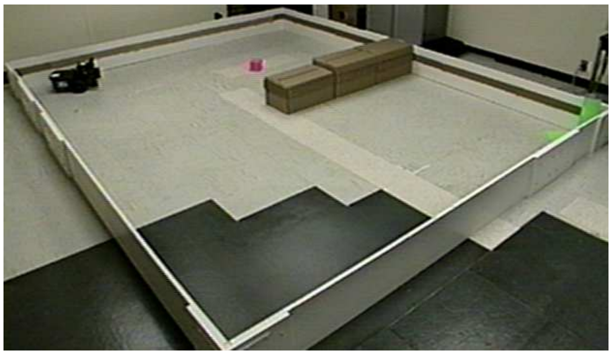


Figure 4: Environment configuration with the pen divided into two regions.

Even in those simpler static conditions (Michaud & Matarić 1997), unexpected behavior selection was learned by the robot. For example, in the configuration with a single block placed at the center of the pen, the robot learned to use *Rest* in front of a wall in order to rotate more than when the *Avoidance* behavior was exploited right away, helping it see the block directly from that location. We were surprised to find that *Turn-randomly* was not frequently useful: at first, we thought that this behavior would help the robot move toward the center of the pen to see the block. Instead, because this behavior made the robot rotate by a random angle at a random interval, it was not very reliable in helping the robot see the block. In fact, it frequently made the robot loop around the block. It is important to note that all of these discoveries were made by the robot from its own point of view, instead of by the designer.

We also performed experiments by starting with one block placed in the center (for the first 60 trials), and then later dividing the pen into two regions (shown in Figure 4). Figures 5 and 6 show the plot of $E(eval)$ at the first junction of the tree and the number of nodes in the tree for the learning trials of one such experiment. In the first 60 trials, the robot learned to use the *Follow-side* behavior to follow the wall to a corner, and then go to the block. Then, a two-phase change occurred: the strategy changed to using the *Observe* option for a while, and then *Rest* from trial 87 on. This can be seen from the changes in $E(eval)$. In fact, *Observe* and *Rest* were the second and third best options at the first junction of tree before the change of environment configuration occurred. The change of strategies and the stabilization starting around the 87th trial is also corroborated in Figure 6, demonstrating that the algorithm was able to quickly adapt to changes in the environment. In effect, the approach shows transfer of what was learned in the first configuration to what was useful in the new, second configuration: an initial strategy can thus serve as a bias toward the elaboration of a specialized way of behaving, but does not prevent it from changing dynamically.

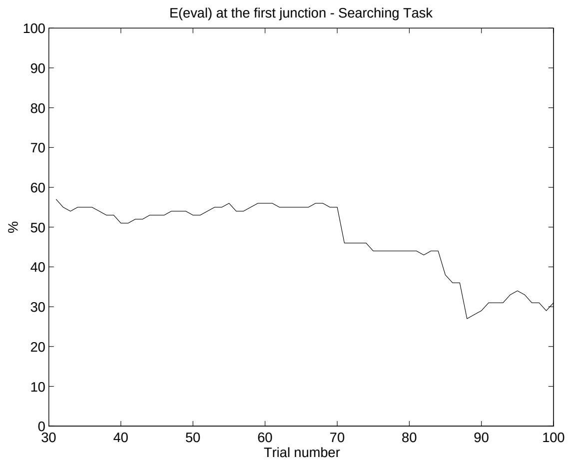


Figure 5: Graph of $E(eval)$ for the Searching Task tree of an experiment starting with one block in the center, and then changing by dividing the pen into two regions.

In the multi-robot experiments, the block to be brought home was put in the center of the pen, as shown in Figure 7. Two or three R1 robots were added to the scenario, and generated interference in the environment. The R1s were programmed to move about, avoid all obstacles and stay out of forbidden regions, using their contact and infra-red

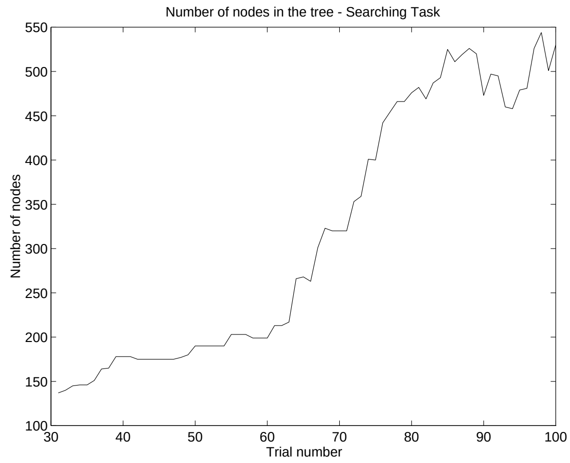


Figure 6: Graph of the number of nodes for the Searching Task tree of an experiment starting with one block in the center, and then changing by dividing the pen into two regions.

sensors, mounted in the front and below their grippers. These forbidden regions were the areas of the pen that were marked with dark tiles or tape, i.e., the area surrounding the block, the home region, and one dark-tiled corner of the pen. The R1 controller was simple: when detecting an obstacle or a forbidden area, the robots moved backward and turned away from the obstacle, then moved forward again; otherwise, it moved forward freely. These constraints on the R1s' motion resulted in a great deal of dynamic interference. Additionally, although dark areas were inaccessible to the R1s, sensor noise enabled them to occasionally enter those areas nonetheless.

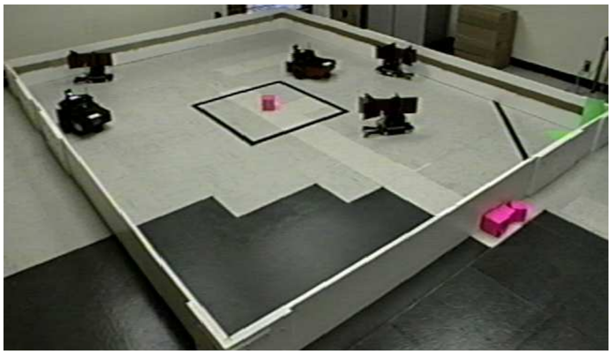


Figure 7: Environment configuration for the multi-robot experiments, with two Pioneers and three R1s. The R1s are programmed to avoid obstacles and areas marked with dark tiles or tape.

Using this configuration, fifteen experiments of 60 to 80 trials were run, lasting between 1.5 and 2 hours and using two or three R1s. In some of these experiments, two learning Pioneer robots were used to compare their learned strategies. However, when placed together, additional sources of noise affected the behaviors of the robots: 1) when facing each other at particular angles, crosstalk between the sonars sometimes occurred; 2) one Pioneer robot searching for a block could perceive and pursue the block pushed by the other; 3) the blue Pioneer robot sometimes confused the red Pioneer robot with a pink block.

During these experiments, the non-stationary conditions arose from the presence and movement of the R1s, and varied greatly during and between experiments. In some cases, one or two R1s stayed close to the home region, while in others all moved around the region near the block. As the learning is primarily dependent on the sequence of past experiences, which changes in each experiment run, optimality cannot be determined from the task alone (unless a very large amount of data is available for statistical analysis). To counter this, we used the run-time data and studied the experiment video tapes in order to analyze what, why, and how strategies were learned, if regularities were found from the interactions dynamics, and how the algorithm adapted to changes detected from these interactions.

For conciseness, we describe only the results obtained for the Searching Task because they best illustrate the properties of the learning algorithm. In this task, the learned strategies largely involved the use of the *Observe*, *Rest* and *Turn-randomly* options. The sequences of options learned were: *Observe – Rest*; *Observe – Rest – Observe*; *Rest*; *Rest – Turn-randomly*; *Rest – Observe*; and *Turn-randomly*. The strategies changed over time according to past experience: when performance for one strategy decreased, another was used based on what was learned previously. One or two stable strategies "won out" in most experiments.

Figure 8 shows an example of a tree learned by the robot after 68 trials. Some nodes were removed for conciseness: the number of nodes removed in a path is presented in the paths and delimited by periods. $E(eval)_i$ values for the sub-paths at the first junction in the tree are also shown. The *Observe* option (marked with '***') is the one with maximum $E(eval)_i$ at the first junction. Note that this path may or may not be followed depending on $eval(t)$ and the use of the selected option.

Figure 10 shows the trace of the choices made for the first 16 trials and the last 10 trials of a test. Data associated with a choice are delimited by parentheses and consist of the trial number, the name of the option chosen, the name of the selection criterion used, $eval(t)$ and $E(eval)$ at the position of the tree where the choice was made. In the initial trials, *Follow-side* was the first favored option. However, this strategy did not remain stable for long; from trials 28 to the end of the experiment, the *Observe – Rest* strategy was preferred (e.g., in trials 58, 62, 63, 65 to 68). Between these trials, different options were explored. These choices were made using the TM and GOT criteria. Consecutive choices with the same $E(eval)$ values indicate that they were made from the same position in the tree. Notice that for trials 62, 64 and 66, a choice using the TM criterion is overridden by one made using the UO or the GOT criterion because $eval(t)$ fell below $E(eval)$. Shorter paths are a sign that the robot was able to find the block without experiencing much interference from the R1s. Otherwise, longer sequences of choices were generated because longer paths were reused. From Figure 8, it is also possible to see

```
((S 41)
 ((A 39)
  ((S 35)                                          ;E(eval)=50 ***
   ((A 23)
    ((R 21)                                        ;Second choice
     ((A 14)                                       ;Case 1
      ((S 8) (A 8)
       ((S 6)
        ((E 2 50) )                                ;Best of Case 1
        ((A 4)
         ((S 2)
          ((E 1 49) )
          ((A 1) .3. (E 1 9)))
         ((R 2) (S 2) (A 2) (S 2) (A 2)
          ((S 1) (E 1 35))
          ((F 1) (S 1) .6. (E 1 0)))))
       ((TR 1) (S 1) .12. (E 1 0))
       ((R 1) (S 1) .4. (E 1 0)))
      ((TR 3) (S 3) (A 3) (S 3) (A 3)
       ((E 1 6) )
       ((S 1) (A 1) .3. (E 1 0))
       ((R 1) (S 1) (A 1) (E 1 0)))
      ((F 3)
       ((A 1) (S 1) .4. (E 1 59))
       ((S 2) (A 2) (S 2) (A 2) (S 2) (A 2) (S 2)
        ((E 1 0))
        ((A 1) (S 1) .10. (E 1 0)))))
     ((S 7)                                        ;Case 2
      ((E 4 92))                                   ;Best of Case 2
      ((A 3)
       ((S 2) (A 2) .18. (S 2)
        ((E 1 0))
        ((A 1) (S 1) .7. (E 1 0)))
       ((E 1 54)))))
    ((TR 1) (S 1) (A 1) .32. (E 1 0))
    ((F 1) (S 1) (A 1) (E 1 8)))
   ((E 12 89)))
  ((F 1) (S 1) .5. (E 1 42))                       ;E(eval)=42
  ((TR 3) (S 3)                                    ;E(eval)=43
   ((A 2) (S 2) (A 2)
    ((S 1) (A 1) (S 1) (A 1) (E 1 55))
    ((F 1) (A 1) (S 1) (A 1) (S 1) (E 1 18)))
   ((E 1 58))))
 ((E 1 100)))
```

Figure 8: Searching Task tree for a multi-robot experiment.

that the algorithm allowed the robot to discriminate two situations using the *Observe – Rest* strategy: using *Rest* in front of a static obstacle (Case 1) and using *Rest* in front of a moving obstacle (Case 2). This distinction is made according to the behavior used after resting, without using any special processing to distinguish a moving from a static obstacle, and lead to distinct strategies for behavior selection. Finally, Figure 9 shows that the number of nodes stopped increasing and started to decrease, demonstrating reuse of stored paths in the tree, even in non-stationary conditions.

Finally, when two Pioneer robots were put in the pen at the same time, they specialized differently in their ways of accomplishing the tasks, based on their specific experiences. Having two Pioneers or more R1s in the pen did not influence the Pioneers' ability to find stable strategies; the dynamics of the interaction with the environment had a more significant effect. Thus, dynamic specialization can occur without requiring complex communication and arbitration schemes for managing the group.

Overall, the results obtained show that the different components of the learning algorithm try to establish a compromise between *exploration* (learning to adapt to

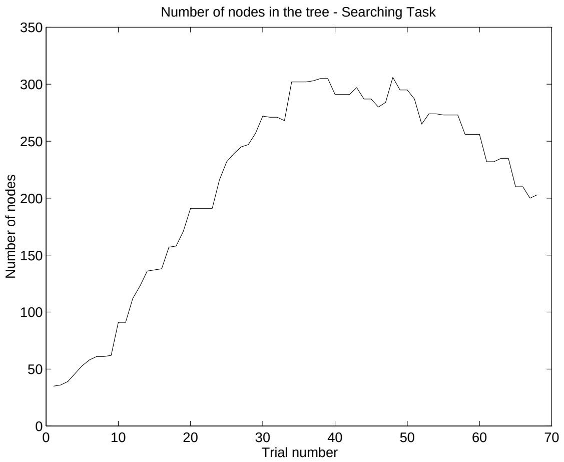


Figure 9: Graph of the number of nodes in a Searching Task tree in a multi-robot experiment.

noise and changes in the environment) and *exploitation* (Kaelbling, Littman & Moore 1996) of a stable behavior selection strategy. The evaluation function characterizes the current situation and past experiences. The tree representation captures sequences of behavior use in a compact fashion to make a decision based on past experiences. The recommendation criteria exploit both of these components to determine when and what choice to make: the TM criterion favors exploitation of the options, while the UO criteria favor exploration for the purpose of finding a different best option. The GOT plays an intermediate role between exploration of options at a given node (when the algorithm does not know which choice to make), and exploitation of the option with the best performance. It serves as a *medium-term memory* of the performances of the options. At the same time, the GOT makes the learning algorithm adapt more easily to changes in the environment that can be reflected in a more constant fashion in the performance of the trials. For a frequently used strategy, $E(eval)$ is more difficult to change because its evaluation is made according to a larger number of trials. In contrast, the GOT is not subject to this and allows for faster exploration by exploiting outcomes of recent trials. The GOT criterion is preferred to random behavior selection because it favors using knowledge derived from past experiences instead of blindly exploring all of the options. Finally, path deletion influences the choices made by affecting $E(eval)$ in the tree. The deletion factor defines the memorizing ability of the algorithm, which can also be viewed as a principled bias set up by the designer. It does not affect the type of strategy learned, but rather how it is learned.

In effect, these influences allow the algorithm to establish *critical points* for behavior selection, i.e., decision factors for recommending the activation of behaviors. These critical points are self-determined and modified in response to continuing changes in the dynamics, which could be caused by alterations of the environment, the behavior of other agents, or the variability of the agent's abilities (e.g., due to noise, battery power, etc.). Even in a non-stationary environment, these critical points provide the agent with some ability to make expectations based on previous experiences, and thus evaluate its current performance.

```
1:(O TM 0 0)
2:(S TM 98 17)
3:(F UO 0 40)
4:(R UO 0 60) (O TM 98 0)
5:(TR UO 0 58) (O TM 96 0)
6:(F TM 75 57)
7:(F GOT 0 62) (S TM 96 91) (F UO 89 91)
8:(F TM 90 63)
9:(F TM 75 65)
10:(F TM 95 66) (TR UO 70 76) (O TM 42 0)
11:(F GOT 0 61) (F UO 87 91)
12:(F TM 96 62) (O TM 64 0)
13:(F TM 96 57) (R UO 45 64) (O TM 43 0)
14:(F TM 92 53) (F UO 45 82) (O TM 43 0)
15:(S GOT 0 50) (S TM 98 17)
16:(TR TM 96 49)
...
58:(S GOT 0 49) (R TM 92 37) (F TM 96 32) (O TM 53 0)
59:(TR TM 92 49)
60:(S GOT 0 50)
61:(TR TM 94 51) (S TM 77 55) (F UO 54 55)
62:(S TM 98 52) (S GOT 51 52) (R TM 58 37) (S TM 57 29)
   (R UO 26 32) (O TM 8 0)
63:(S TM 97 50) (R TM 89 35) (F TM 88 27) (O TM 13 0)
   (O TM 13 0)
64:(S TM 87 50) (64 S GOT 37 50)
65:(S GOT 0 50) (R TM 88 34) (E TM 96 27) (O TM 92 0)
   (O TM 0 0)
66:(S TM 96 50) (S GOT 48 50) (R TM 56 32)
67:(S GOT 0 51) (R TM 99 34) (S TM 97 25) (S TM 83 27)
   (R TM 57 31) (S TM 65 35) (F UO 34 35) (O TM 3 0)
68:(S TM 98 51) (R TM 75 33) (TR TM 74 23) (E TM 66 3)
   (R UO 2 3) (O TM 0 0)
```

Figure 10: Trace of choices made for the Searching Task tree in a multi-robot experiment.

5 Related work

The described approach shares similarities with issues and methodologies from the field of machine learning, principally the reinforcement learning paradigm. In our learning algorithm, the decision is derived from stored past experiences and the current situation in the tree; an action takes the form of a behavior recommendation, and the reward is based on the internal measure of performance, which is also deduced from behavior use over time. A key difference from the commonly used reinforcement learning algorithms is that we did not use a scalar reinforcement signal, or one completely determined by perceived conditions.

As in Dorigo & Colombetti (1994), Asada *et al.* (1994), and Matarić (1997*b*), our algorithm learns at the behavior gating level (i.e., decides which behavior's action should be switched through and executed (Kaelbling et al. 1996)), and the main difference from the above approaches is in our use of history of behavior use instead of percepts as states for selecting behaviors. Taking into consideration the history of past experiences was inspired by McCallum (1996), whose approach partitions the state space from raw sensory experiences, and uses memory of features encoded in a U-tree to augment the agent's perceptual inputs. Globally, it learns a variable-width window that serves simultaneously as a model of the environment and a finite-memory policy (Kaelbling et al. 1996). In our case, the algorithm learns a finite-memory strategy of behaviors use and selection, since it uses behaviors as an abstraction. Learning from observed behavior activation was also explored within the case-based reasoning framework, in work by Ram & Santamaria (1993). However, the computational complexity of their approach is much greater than ours.

Finally, the development of our algorithm was done in the context of a general control architecture for an intelligent agent (Michaud et al. 1996) based on dynamic selection of behaviors.

6 Conclusion

Learning in dynamic and unpredictable environments, as in the multi-robot domain, is a very challenging problem. The goal of our approach is to make a robotic agent learn from self-evaluation of its behavior in order to compute, represent, and utilize the interaction dynamics with its environment. The algorithm uses history of behavior use to derive a continuously and incrementally updated tree representation as a model of these interactions. Learning the model and using it to derive behavior selection strategies is done relative to the exploitation of the controlling resources (i.e., the behaviors) of the agent, and thus defined according to the agent's own perspective. This property decreases the burden of behavior design by not having to fine-tune the behaviors or acquire knowledge about the specific and complex dynamics of the system. The approach is computationally efficient and the demonstrated learning is performed on-board the robot in real-time in dynamic and noisy domains.

Results show the effectiveness of the approach in taking advantage of any regularities experienced in the world, leading to fast and adaptable agent specialization, and generating when possible temporarily-stationary conditions in a multi-robot environment. Since interaction with the environment is affected by the dynamics and the agent's own limitations in perception, action, and processing, our approach advocates internal evaluation of the interaction dynamics, and taking advantage of them while they are still valid. By doing so, the approach leads to learning unanticipated behavior strategies from the perspective of the agent instead of the designer, thus increasing agent autonomy. It also provides a predictive capability that is typically lacking in behavior-based systems, thus making a move toward increasingly "cognitive" means for generating complex behavior.

Acknowledgments

Support for the Postdoctoral Fellowship of F. Michaud was provided by NSERC (Natural Sciences and Engineering Research Council of Canada). This research is also funded by the Office of Naval Research Grant N00014-95-1-0759, the National Science Foundation Infrastructure Grant CDA-9512448 and the Career Grant IRI-9624237 to M. J. Matarić. The authors also want to thank P. Melville, B. Werger, D. Goldberg and the rest of the Interaction Lab for their insightful comments.

References

Agre, P. E. (1988), The Dynamic Structure of Everyday Life, PhD thesis, MIT.

Asada, M., Uchibe, E., Noda, S., Tawaratsumida, S. & Hosoda, K. (1994), Coordination of multiple behaviors acquired by a vision-based reinforcement learning, *in* 'Proc. IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems', Munich, Germany.

Brooks, R. A. (1986), 'A robust layered control system for a mobile robot', *IEEE Journal of Robotics and Automation* **RA-2**(1), 14–23.

Brooks, R. A. (1991), 'Intelligence without representation', *Artificial Intelligence* **47**, 139–159.

Brooks, R. A. (1996), MARS: Multiple Agency Reactivity System, Technical report, IS Robotics.

Dorigo, M. & Colombetti, M. (1994), 'Robot shaping: Developing autonomous agents through learning', *Artificial Intelligence* **71**(4), 321–370.

Goldberg, D. & Matarić, M. J. (1997), Interference as a tool for designing and evaluating multi-robot controllers, *in* 'Proc. National Conf. on Artificial Intelligence (AAAI)', Providence, RI, pp. 637–642.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), 'Reinforcement learning: A survey', *Journal of Artificial Intelligence Research* **4**, 237–285.

Maes, P. (1989), The dynamics of action selection, *in* 'Proc. Int'l Joint Conf. on Artificial Intelligence (IJCAI)', Detroit, MI, pp. 991–997.

Matarić, M. J. (1997*a*), 'Behavior-based control: Examples from navigation, learning, and group behavior', *Journal of Experimental and Theoretical Artificial Intelligence*.

Matarić, M. J. (1997*b*), 'Reinforcement learning in the multi-robot domain', *Autonomous Robots* **4**(1).

McCallum, A. K. (1996), Learning to use selective attention and short-term memory in sequential tasks, *in* P. Maes, M. J. Matarić, J.-A. Meyer, J. Pollack & S. W. Wilson, eds, 'From Animals to Animats: Proc. 4th Int'l Conf. on Simulation of Adaptive Behavior', The MIT Press, Cape Cod, pp. 315–324.

Michaud, F. & Matarić, M. J. (1997), A history-based learning approach for adaptive robot behavior selection, Tech. Report CS-97-192, Computer Science Department, Volen Center for Complex System, Brandeis University, Waltham, MA.

Michaud, F. & Matarić, M. J. (1998), 'Learning from history for behavior-based mobile robots in non-stationary conditions', to appear in the joint special issue of Machine Learning and Autonomous Robots on "Robot Learning", H. Hexmoor and M. Mataric, eds.

Michaud, F., Lachiver, G. & Dinh, C. T. L. (1996), A new control architecture combining reactivity, deliberation and motivation for situated autonomous agent, *in* P. Maes, M. J. Matarić, J.-A. Meyer, J. Pollack & S. W. Wilson, eds, 'From Animals to Animats: Proc. 4th Int'l Conf. on Simulation of Adaptive Behavior', The MIT Press, Cape Cod, pp. 245–254.

Ram, A. & Santamaria, J. C. (1993), 'Multistrategy learning in reactive control systems for autonomous robotic navigation', *Informatica* **17**(4), 347–369.