A reactive navigation method based on an incremental learning of tasks sequences

Frédéric Davesne, Claude Barret

A Reactive Navigation Method Based on an Incremental Learning of Tasks Sequences

Frédéric Davesne
CEMIF-CSG-University of Evry
40, r. Pelvoux, 91020 Evry Cedex
davesne@cemif.univ-evry.fr

Claude Barret
CEMIF-CSG-University of Evry
40, r. Pelvoux, 91020 Evry Cedex
cbarret@cemif.univ-evry.fr

Abstract

Within the context of learning sequences of basic tasks to build a complex behavior, a method is proposed to coordinate a hierarchical set of tasks. Each one possesses a set of sub-tasks lower in the hierarchy, which must be coordinated to respect a binary perceptive constraint. For each task, the coordination is achieved by a reinforcement learning inspired algorithm based on an heuristic which does not need internal parameters. A validation of the method is given, using a simulated Khepera robot. A goal-seeking behavior is divided into three tasks: go to the goal, follow a wall on the left and on the right. The last two ones utilize basic behaviors and two other sub-tasks: avoid obstacles on the left and on the right. All the tasks may use a set of 5 basic behaviors. The global goal-seeking behavior and the wall-following and the obstacle avoidance tasks are learned during a step by step learning process.

1 Introduction.

1.1 Development context.

Within the framework of mobile robotics, it is often difficult to establish a relationship between the data perceived by the robot and the behavior it must achieve according to its input data.

Indeed, the perceptive data may be very noisy or may not be interpreted easily, so that modeling the mapping between perception and behavior could be a very difficult task. Reinforcement learning methods [1] have been widely used in that context [2],[3], mainly because they do not need a prior knowledge about the process model. Moreover, they theoretically achieve incremental learning and they can cope with a possible inertia of the system. But finding suitable internal parameters for those algorithms is not intuitive and may be a difficult task [4]. Besides, it is not easy to find a compromise between the stability and the robustness of the algorithm and its incremental characteristic. Finally, given that the reinforcement methods need to sufficiently explore the perception space before finding a quite good solution, learning to fit a complex behavior in a reasonable lapse of time turns to be impossible without finding out some characteristics of the process, leading to a problem with a significantly decreased perception space. A solution could be to divide the whole task into coordinated sub-tasks [5], each one being easier to learn than the complex behavior. However, the problem is turned into another one: choosing to execute a precise sub-task is often tricky, especially if the choice depends on the perceptual data of the agent. In that case, applying a simple switching is not generally sufficient; the agent has to learn to decide which sub-task is to be executed according to its input data. Moreover, when a failure in the learning process occurs, one has to know if the cause of the mistake is due to a misleading choice of a sub-task or to an internal deficiency of the elected sub-task unit. In the last eventuality, it could be necessary to modify this unit to make it avoiding the same mistake. So, it must have the capacity to learn at anytime it is used: this is an important focus of incremental learning methods.

1.2 Characteristics of the algorithm.

The proposed algorithm has some hard links with the reinforcement learning concept: it is a trial/failure method, it does not need a prior knowledge of the process model, it copes with the temporal credit assignment problem and it is incremental. However, it is not based on an optimization method, but on the respect of binary perceptive constraints.

The framework of the algorithm is the collective learning of a hierarchical set of tasks. Each one can use a fixed number of tasks lower in the hierarchy. For each behavior which is to learn, the quality of the obtained sequences is given by a binary feedback

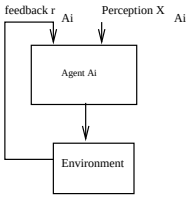

Figure 1: Basic model of the agent-environment interaction.

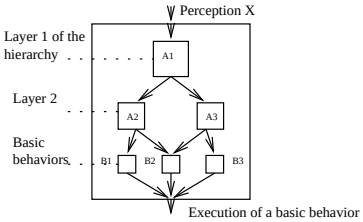

Figure 2: Hierarchical organization of the tasks.

signal. The constraint given by the signal eliminates the sequences that have already lead to a mistaken behavior.

Each task is achieved by an agent (fig. 1) which aim is to choose the sub-task to execute. So, at each time, the master task (highest in the hierarchy) starts the chained decisional process, ending with the execution of a basic behavior (fig. 2).

Each agent has to learn to decide according to its current perception. To do so, an internal representation of the influence of its choices (leading to the execution of a basic task) on its perception is built on-line by the meaning of an oriented graph which nodes are associated with a particular choice and a precise perceptive area. Each node has an internal binary value related to its quality according to the respect of the constraints of the agent. This value is obtained by a back-propagation of the values linked to the ending nodes of the graph (nodes associated with a failure) using a consistency law derivated from the minimax algorithm [6]. Here, an analogy is made with a two player game in which the agent must use some actions so that the response of the dynamic system (its opponent) never leads it to a losing state, that is to say a perceptive state in which the agent does not respect its constraints.

When a failure is given by the feedback signal of an agent whereas the other lower level agents have fulfilled theirs constraints, the first one attributes the mistake to its decision.

1.3 Application.

The algorithm will be applied to a general goal-seeking problem in which the obstacle avoidance is performed by a wall-following behavior. To do so, the mobile robot Khepera [7] simulator written par O.Michel [8], running on Unix-like operating systems, will be utilized, which allows to test the robustness of the algorithm in a very noisy perceptive data context. The incremental capability and the possibility to resolve on-line the conflicts between the coordinated tasks will bee shown.

2 Description of the algorithm.

2.1 Context and notations.

The set of learning agents $S_A = \{A_1, A_2, \ldots, A_m\}$ receives a data stream which can be modeled by a vector $X = \left(X^1, X^2, \ldots, X^n\right)$; each agent may use the whole or a limited set of the elements of X; this sub-set is noted X_{A_i}. Each element of this vector is a bounded value. Besides, the set of agents may use a fixed set of basic behaviors $B = \{B_1, B_2, \ldots, B_p\}$. At each time, the result of the decision taken by the master agent A_1 is the final execution of an element of B.

For each agent A_i, a prior knowledge is introduced before the learning process, which fixes the set of choices (call another agent or execute a basic task) $S_{A_i} = \left\{C_i^1, \ldots, C_i^l\right\}$ the agent can make to fulfill its constraints given at each time by a binary feedback signal r_{A_i} . Each decision C_i^k of S_{A_i} is associated to a branching priority P_i^k, which is a positive integer.

The learning process of A_i consists of an on-line building of a mapping between the X_{A_i} data and the elements of S_{A_i}. To do this, A_i utilizes perceptual areas Z_i^k , each one linked to an element C_i^k of S_{A_i}.

Each Z_i^k is a continuous space generated par the whole set of possible $X_{C_i^k}$. It is divided into a set of boxes $Box_{i,k}^{j,\, j \in \{1,\ldots,b\}}$ made up accordingly to the following set of equations:

$$\begin{cases} \forall k \in \{1,\ldots,l\} \bigcup_{j \in \{1,\ldots,b\}} Box_{i,k}^j = Z_i^k \\ \forall k \in \{1,\ldots,l\} \bigcap_{j \in \{1,\ldots,b\}} Box_{i,k}^j = \emptyset \\ Box_{i,k}^j = \left\{ X_{C_i^k} = \left(X_{C_i^k}^1, \ldots, X_{C_i^k}^{n_{C_i^k}}\right) / \right. \\ \quad \left. \forall j \in \left\{1, \ldots, n_{C_i^k}\right\}, m_{i,k}^j \leq X_{C_i^k}^j < M_{i,k}^j \right\} \end{cases}$$

Thus, each box $Box_{i,k}^j$ is parameterized by $n_{C_i^k}$ couples of values $(m_{i,k}^j, M_{i,k}^j)$ which are the boundary values for each perceptive signal used when C_i^k is called.

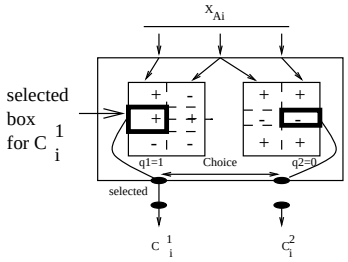

Figure 3: Decision-making of an agent A_i.

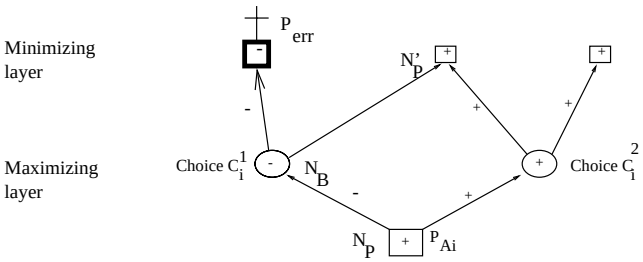

Figure 4: Detail of the graph associated to the learning process of the agent A_i. The current perceptive state is P_{A_i}. The past experience of the agent allows it to detect the transition to 3 different perceptive states from P_{A_i} when the choice C_i^1 is made. The qualities are shown by some + and -, fitting the consistency law derived from the minimax algorithm.

Each box is associated with a binary quality $q_{i,k}^j \in \{0,1\}$, which is the estimated quality of the decision made by the agent A_i to call C_i^k.

Having received the set of the l signals $X_{C_i^k}$, A_i can know precisely which boxes $Box_{i,k}^j$ are fired, therefore which quality $q_{i,k}^j$ is associated to each one of the possible choices. So, the decision linked to the best quality can be taken (fig. 3).

2.2 Modeling the perceptive graph.

The objective is to memorize the sequences of boxes fired in the perceptual area of A_i while it is executing some sub-tasks. The set of boxes fired at each time, associated to the possibles choices, can be seen as a perceptive state: as long as the set of elected boxes does not change, the agent is considered to be in the same state $P_{A_i}(X_{A_i}(t)) = (j_1, \ldots, j_l)$ where each j_k is an integer which is defined by the relationship $X_{C_i^k} \in Box_{i,k}^{j_k}$. If a failure is detected by the way of r_{A_i}, the agent moves in the special state P_{err}. A perceptive graph models the sequences of states. When the agent reaches a perceptive state it has not experienced before in its learning process, a node N_P associated to that state is created. This node is linked to other nodes N_B, which represents the different possible choices that the agent can make among the elements of S_{A_i}. When a decision is taken by the agent, its state is turned into the transitory state associated to the choice it has made. Then, the reaction of the dynamics of the system when executing C_i^k modifies the perception X_{A_i} , allowing the agent to move to another perceptive state at time t' (arc linking N_B and N'_P) (fig. 4).

2.3 Consistency law.

Each node of the perceptive graph is associated to a binary quality. The quality of a state expresses the ability of the agent to avoid moving to the ending state P_{err}. Indeed, the problem which consists on taking a decision according to the fulfillment of constraints with a given dynamics of the system may be seen as a two players game: the agent A_i and the dynamics. The aim of the agent is to never reach the losing state P_{err} , which quality $q_{P_{err}}$ is 0. To do so, we use the minimax algorithm: the minimax searching tree is the perceptual graph and the evaluation function values are the qualities.

The consistency law applied for each node of the graph is given by the following two relationships. The first one is dealing about the perceptive nodes N_P , whereas the second one is applied to the transitory nodes N_B associated to a decision-making when the perceptive state of the agent is N_P:

$$q_{N_P} = \max_{N_i \in Child(N_P)} \{q_{N_i}\} \quad (1)$$

$$q_{N_B} = \min_{N_i \in Child(N_B)} \left\{q_{N'_i}\right\} \quad (2)$$

Where $Child(N_P)$ is the set of the children of N_P in the graph and $Child(N_B)$ is the set of the children of N_B.

2.4 Using the consistency law to learn.

As soon as an arc from a node N_B to a node N'_P is created while the dynamics makes the perceptual data of the agent evolve, $Child(N_B)$ is modified, therefore the consistency relationship could be broken. In that case, the value of the quality of N_B is forced in order to respect the equation 2. If q_{N_B} is modified, the value of the quality associated with the father N_B could be consequently modified due to the equation 1. A sequence of modifications may then happen, leading to a back-propagation of the prior modification. This

ends as soon as the consistency law is fulfilled by the qualities of all the nodes.

2.5 Global learning algorithm.

Step 1- Retrieval of the node N_P linked to the perceptive state of the agent: $P_{A_i}(X_{A_i}(t))$. If it does not exist, create it and create the nodes N_B associated to the choices C_k^i.

Step 2- Decision-making: choice of C_{max} among the elements of S_{A_i}, which associated quality is maximal. If two or more elements of S_{A_i} have the same maximal quality, those which priority is maximal are chosen.

Step 3- Execution of C_{max} (transitory node N_B) and retrieval of the feedback signal r_{A_i}, while the perceptive state remains unchanged and no failure due to A_i is detected. If another agent is utilized, the same algorithm is recursively called for it.

Step 4- Retrieval of the current perceptive node N'_P. If the arc from N_B to N'_P does not exist, create it and use the consistency law on this arc.

At each time, the first call to this algorithm is carried out for the master agent A_1.

3 Experimental results.

3.1 Context of the experiments.

The experiments have been carried out with the help of the Khepera simulator. Khepera (fig. 5) is a small mobile robot developed at Ecole Polytechnique Fédérale de Lausanne (EPFL) which has a circular shape featuring 55 mm in diameter. It possesses 8 infrared sensors $s_1, \ldots, s_8$, allowing the measurement of distances in a short range from about 1 cm to 5 cm and the values they give ranges from 0 (no obstacle found) to 1024 (an obstacle is very near).

The Khepera simulator reproduces the imperfections of the sensors, so that it has been noticed that the experimental results deduced from the real and the simulated Khepera are very close.

In the following experiments, the simulated robot is controlled by receiving the values of the linear speed ls_1 and ls_2 of its two wheels. These values ranges from -10 to 10, corresponding to a maximal speed of about 40 mm/s.

The objective is to build a goal-seeking behavior, making the hypothesis that the absolute coordinates of both the goal and the robot are supposed to be precisely known at each time. The obstacle avoidance

Behavior	meaning	ls_1	ls_2
B_1	Move forward	3	3
B_2	Move to the right	2	0
B_3	Move to the left	0	2
B_4	Turn on the right	2	-2
B_5	Turn on the left	-2	2

Table 1: Basic behaviors utilized in the experiments. The ls_1 and ls_2 values come without any unit.

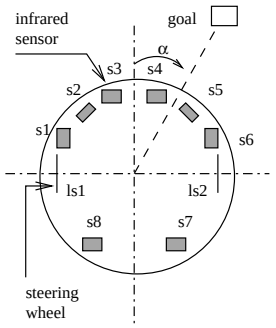

Figure 5: The miniature mobile robot Khepera.

is performed by a wall following behavior, divided into two sub-tasks: follow the wall on the left and follow the wall on the right.

The agent possesses 4 input signals: $X = (d_{left}, d_{forward}, d_{right}, \alpha)$. α is the angle between the robot direction and the goal. The value of the angle is supposed to be known at each time. $d_{left} = \max(s_1, s_2)$, $d_{forward} = \max(s_3, s_4)$, $d_{right} = \max(s_5, s_6)$

Five basic behaviors have been chosen, which are linked to a couple (ls_1, ls_2): $B = \{B_1, B_2, B_3, B_4, B_5\}$. Their specification is given by table 1.

The robot possesses three internal binary feedback signals and an external one: BUMP, FWL and FWR plus BLK. BUMP is equal to 1 if the robot has bumped into an obstacle, else it is equal to 0. FWL (resp. FWR) is equal to 0 if the d_{left} (resp. d_{right}) value has remained smaller 10 for more than 30 learning steps; else, it is equal to 1. BLK is equal to 1 when a sequence of conflicting behaviors is detected ("turn on the left" followed by "turn on the right" is an example). BLK is sent by the user.

3.2 Learning protocol.

The global learning process is divided into two stages: the wall-following behaviors learning and the context switching learning. During the first one, four agents are trained: A_1, A_2, A'_1 and A'_2 (fig. 6).

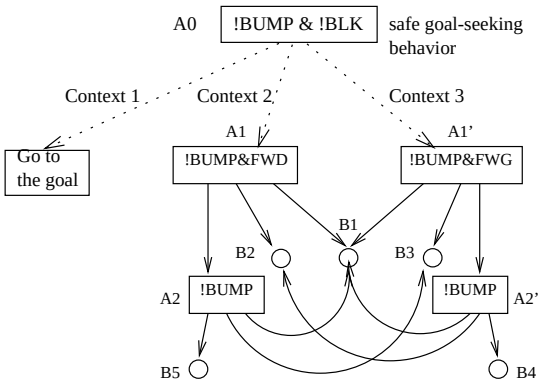

Figure 6: Hierarchical set of agents used by Khepera. The constraints are combinations of BUMP, FWD, FWG and BLK.

Each of these agents are supposed to be context-free: when a failure occurs after one has taken a decision, it is responsible of the mistake whatever its perception is. During the second stage, the switching between the three contexts is learnt by A_0; "go to the goal" has the highest priority. The agents A_2 and A_2' are trained independently before A_1 and A_1'. A training period is considered successful when 500000 consecutive learning steps are done without a failure. At the beginning of the A_2 and A_2' learning processes, the robot is initialized in the free space of the environment (fig. 7). At the beginning of the A_1 and A_1' learning processes, the robot is initialized near a wall. The perceptive areas associated to these four agents are three dimensional continuous spaces generated by $(d_{left}, d_{forward}, d_{right})$; they are regularly divided into $4 \times 4 \times 4 = 64$ boxes. But, the agent A_0 has a four dimensional space generated by $(d_{left}, d_{forward}, d_{right}, \alpha)$, divided into $4 \times 4 \times 4 \times 4 = 256$ boxes.

3.3 Results.

- First learning stage
  10 learning attempts have been done for A_2. All the attempts were successful, ending after 32 up to 56 trials. Fig. 8 shows the evolution of the number of consecutive learning steps without failure. It is noticed that the duration of a trial is a function of the number of nodes (fig. 9). It simply means that as soon as the agent has a wide perceptive experience, it is able to respect its constraints with making very few mistakes. According to the perceptive areas division process, the maximal number of nodes for all the agents is $64 + 3 \times 64 + 1 = 257$. This number is nearly reached at the end of the learning stage. As a consequence, we have noticed that the failures that come within the learning process of master agent A_1 using A_2 are not the result of a mistake made by A_2: they all are due to misleading decisions taken by A_1. The learning period of A_1 has successfully ended after 20 up to 25 trials.

- Second learning stage
  The main issue of this stage is that using the three tasks may generate sequences of conflicting subtasks, such as "move to the left" and "move to the right". But, it can be much more complicated than this example. However, this situation always leads to a cyclic sequence of perceptual states in A_0. So, when the robot is trapped in such a sequence of tasks, a BLK signal is sent by the user. Then, if it received while context 1 is running ("go to the goal"), the choice of context 1 is punished. If a conflict arises between the two wall-following behaviors, a BLK signal leads to punish the left wall-following if the right side of the robot is nearer of an obstacle than the left side. This strategy permits to learn to avoid conflicts, but the learning process is longer than in the first stage. Indeed, although the number of possible perceptive states is small for A_0, a lot of conflicts happen, so that it takes about 300 trials to make the robot reaching the goal without a failure (fig. 7).

4 Discussion and future work.

The learning method and the algorithm allow to incrementally learn a hierarchical set of coordinated tasks. Each task is considered to be a sequence of sub-tasks lower in the hierarchy. During the learning process, a binary constraint is used to eliminate the sequences that do not fit the objective of the task. A binary constraint permits to eliminate the sequences of sub-tasks that do not fit the objective of the task. Each behavior is managed by an agent which uses an internal perceptive graph to memorize the dynamics of the system in order to predict the consequences of the execution of each sub-task. The nodes of the graph possess a binary quality which is obtained by a back-propagation of the terminal node quality, using a consistency law inspired by the AI minimax heuristic.

Utilizing a simulated Khepera robot which aim is to learn a safe goal-reaching behavior, it has been shown that the hierarchical set of agents created for this be-

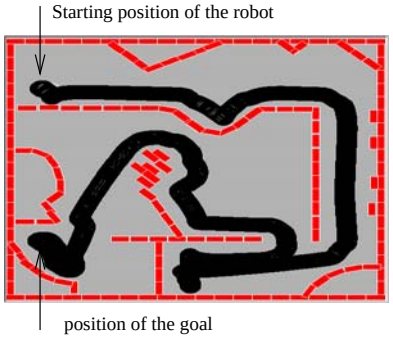

Figure 7: Goal-reaching behavior in the Khepera simulator environment.

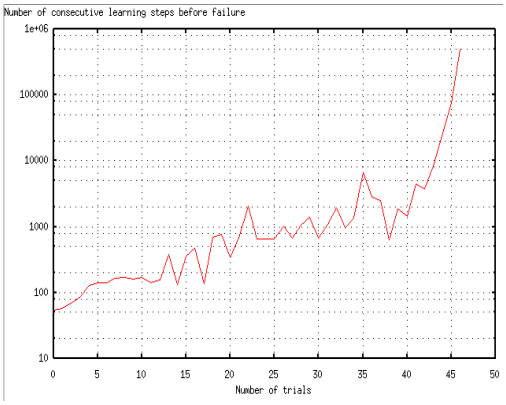

Figure 8: Evolution of the number of consecutive learning steps without a failure of the agent A_2.

havior are able to learn at any time they are utilized. Thus, although they principally learn during their own learning stage, they can evolve if they are used in a high level algorithm framework where they have to face new perceptive situations. Moreover, the algorithm can cope with very noisy perceptive data produced by the infra-red sensors of Khepera. Besides, the system can learn to avoid the conflicts between the wall-following and goal-reaching tasks.

References

[1] C.J.C.H. Watkins. "Learning from delayed rewards". PhD thesis, King's College, University of Cambridge, May 1989.

[2] L-J. Lin. "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching". In Machine Learning, (8), pp.293-321, 1992.

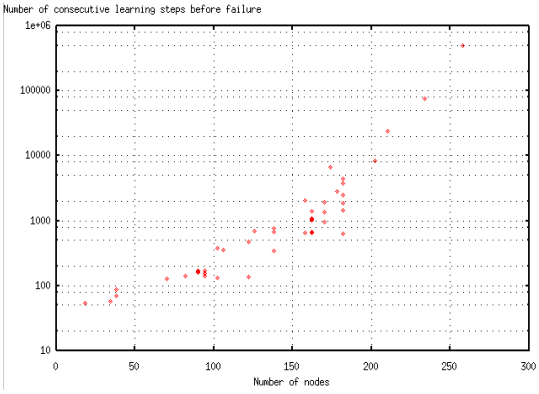

Figure 9: Relationship between the number of nodes within the perceptive graph of A_2 and the number of consecutive learning steps without a failure.

[3] M.Asada, S.Noda and K.Hosoda. "Action-Based Sensor Space Categorization for Robot Learning". In IROS 96, pp1502-1509, 1996.

[4] H.Bersini and V.Gorrini. "Three Connectionist Implementations of Dynamic Programming for Optimal Control: A Preliminary Comparative Analysis". In Workshop on NN for Id. Ctrl. Rob. , 1996.

[5] M.Asada, E.Uchibe, S.Noda, S.Tawaratsumida and K.Hosoda. "Coordination of Multiple Behaviors Acquired bya Vision-Based Reinforcement Learning". In IROS 94, pp917-924, 1994.

[6] E.Rich, "Artificial Intelligence". McGraw-Hill, 1983.

[7] F.Mondada, E.Franzi and P.Lenne "Mobile robot miniaturisation: a tool for investigation in control algorithms". In Proc.3. Int. Symposium on Experimental Robotics, Kyoto, 1993.

[8] O.Michel. Khepera Simulator Package version 2.0: Freeware mobile robot simulator written at the University of Nice Sophia-Antipolis. Downloadable at http://wwwi3s.unice.fr/~om/khep-sim.html , 1996.