A Cohenence Approach to Learning from Reward. Application to the Reactive Navigation of a simulated Mobile Robot

Frédéric Davesne, Claude Barret

Constraint based Memory Units for Reactive Navigation Learning

Frédéric Davesne* **and Claude Barret****

**CEMIF - Systèmes Complexes, 40, Rue du Pelvoux - 91020 EVRY CEDEX - FRANCE, davesne@cemif.univ-evry.fr*

*** CEMIF - Systèmes Complexes, 40, Rue du Pelvoux - 91020 EVRY CEDEX - FRANCE, cbarret@cemif.univ-evry.fr*

Abstract. Within this paper, a new kind of learning agents - so called Constraint based Memory Units (CbMU) - is described. The framework is the incremental building of a complex behaviour, given a set of basic tasks and a set of perceptive constraints that must be fulfilled to achieve the behaviour; the decision problem may be non-Markovian. At each time, one of the basic tasks is executed, so that the complex behaviour is a temporal sequence of elementary tasks.
A CbMU can be modelled as an adaptive switch which learns to choose among its set of output channels the one to be activated (given its perceptive data and a short term memory), in order to respect a particular constraint. An output channel may be linked either to the firing of a basic task or to the activation of another CbMU; this allows a hierarchical decisional process, implying different levels of contexts.
The dynamics of the system is learnt by the mean of a perceptive graph and the cycles detected by the short term memory of a CbMU are utilised as sub-goals to build internal contexts. The learning procedure of a CbMU is a reinforcement learning inspired algorithm based on an heuristic which does not need internal parameters. It is achieved by a consistency law between the binary values of the connected nodes of the perceptive graph, inspired from the AI minimax algorithm.
In this article, an example of programming with CbMUs is given, using a simulated Khepera robot. The objective is to build a goal-reaching behaviour which is formulated by a high level strategy composed of logical rules using perceptive primitives. Four CbMUs are created, each one dedicated to the exploitation of particular perceptive data, and five basic tasks are utilised.

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 behaviour it must achieve according to its input data.
Indeed, the perceptive data may be very noisy or may not be interpreted easily, so that modelling the mapping between perception and could be a very difficult task. Reinforcement learning methods (Watkins, 1989) have been widely used in that context (Lin, 1992),(Asada et al., 1996), 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 (Bersini and Gorrini, 1996). Besides, it is not easy to find a compromise between the stability and the robustness of the algorithm and its incremental characteristic. So, the learning stage may be fast, but the amount of time needed to develop a successful experiment is often important. Finally, given that the reinforcement methods need to sufficiently explore the

perception space before finding a suitable solution, learning to fit a complex behaviour 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, each one being easier to learn than the complex behaviour. 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 Overview.

The framework is the incremental building of a complex behaviour, given a fixed set of basic tasks. We suppose that the desired task can be seen as a set of constraints. For example, the cart pole problem (fig. 1) possesses two constraints which must be verified at each time: $X \in [X_{min}, X_{max}]$ and $\theta \in [\theta_{min}, \theta_{max}]$.
So, a decisional process must be learnt, according to the perceptive constraints, in order to fulfil them at each time. A CbMU is a part of the decisional process. It is an adaptive switch which learns to choose among its output channels the one to be activated, given its input data. Here, the learning criteria is the respect of the CbMU constraint. Thus, it differs from the typical reinforcement based methods, whereas it has some hard links with the reinforcement learning concept: it is a trial/failure method which does not need a prior knowledge of the process model and it is incremental.

1.3 Validation of the computing method that uses CbMUs.

A general goal-seeking problem will be computed, in which the obstacle avoidance is performed by a wall-following behaviour. To do so, the mobile robot Khepera (Mondada et al., 1994) simulator written by O.Michel (Michel, 1996), running on Unix-like operating systems, will be utilised, which allows to test the robustness of the algorithm in a very noisy perceptive data context. The incremental capability and the learning rapidity of the algorithm will be shown.

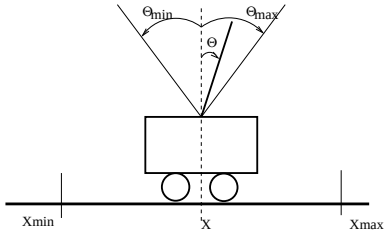

Figure 1: The cart pole problem: a typical constraint based issue.

2 Constraint based Memory Unit specifications.

2.1 Main ideas

framework In this paper, we suppose that the task which is to learn can be achieved with a temporal sequence of a finite set of basic tasks (let p be the number of the basic tasks). Thus, at each time, one of them is executed, in order to fulfil a set of binary constraints. A constraint K can be written like this: $\forall t, X_{min} < X(t)$ or like this: $\forall t, X(t) < X_{max}$where X is one signal of the continuous input space of the learning agent.
hierarchical decomposition of the task Let consider a very simple task (T): "follow a wall", which is carried out with three basic tasks: "go forward", "move on the left" and "move on the right". The task can be divided into "follow a wall on the left"(T_1) OR "follow the wall on the right"(T_2). T_1 can be expressed like this: "do not bump into a wall on the left"(T_3) AND "do not be too far from the wall on your left"(T_4). The same decomposition can be done for (T_2). The choice between T_1 and T_2 is context-dependant; one decides to execute one of the two sub-tasks depending on two different contexts: "there is a wall on the left and there is no obstacle on the right"(C_1) and "there is a wall on the right and there is no obstacle on the left"(C_2); the choice between T_3 and T_4 is also context-dependant: "Am I going to bump into the wall ?"(C_3) and "Am I going to be too far from the wall ?"(C_4). All the contexts can be expressed with constraints.
We notice that the choice among the three basic tasks implies a hierarchical decisional process at each time:

(T) [K_T = K_{T_1} OR K_{T_2}] IF CONTEXT(T)=C_1 DO T_1 ELSE IF CONTEXT(T)=C_2 DO T_2 ELSE This is not a proper context for following a wall
(T_1) [K_{T_1}] IF CONTEXT(T_1)=C_3 DO T_3 ELSE IF CONTEXT(T_1)=C_4 DO T_4 ELSE choose the basic task "go forward"
(T_3) Choose the basic task "move on the right"
(T_4) Choose the basic task "move on the left"
K_{T_1} can be expressed with the input signals of the system.
(some identical sub-tasks can be done for (T_2))
This basic example shows that we have built a program with some a priori knowledge upon what we precisely know about the task (for example, "if I am far from the wall on my left side, move on the left"). The bounds of the contexts and the switches from one context to another have to be learnt. This is done by the CbMUs, each one coping with a particular switch ($C_1 \leftrightarrow C_2$,$C_3 \leftrightarrow C_4$). The decomposition may reduce the input space or the output space for each learning switch.
Thus, knowing the hierarchical decomposition of the constraints and the set of basic tasks, the problem is to shape the different contexts and to learn how the dynamics brings the system from a context to another to fulfil the constraints.

Context specification We assume that a context is not reduced to an area of the input space but also includes a short term memory (the task may be non-Markovian). Thus, a decision is taken according to the current input signal and the content of the short term memory.

Coarse description of a CbMU A CbMU has a specific constraint to cope with. Its input space is continuous and is divided into a set of boxes (let n be the dimension of the input space). The CbMU may switch from one sub-task to another one when its input signal moves from one box to another.
The binary constraint of the CbMU is a set of conditions upon some of the components of the input space. For example, in the cart-pole problem, two of the four input components possess a constraint (X and θ).
The CbMU learning process is based on a coarse learning of the dynamics of the system, by the mean of a perceptive graph (fig. 2). Each box of the input space is associated to a precise node (round node). The action of choosing a sub-task when entering a box (when the input signal moves from one box to another) is linked to a square node. The arc from a round node to a square node symbolises the choice of the CbMU whereas the arc from a square to a round node represents the response of the dynamics of the system when having chosen a precise sub-task from a particular box. The node "E" is reached whenever the constraint is not fulfilled.
When entering a box, a sub-task is selected. The decision is taken regarding the binary quality of each action node.

Consistency law Each node of the perceptive graph possesses a binary quality (- or +). At the beginning of the learning process, the quality of each node is +, except the quality of the ending node (-). For we consider the learning of the dynamics as a two players games (the CbMU and the dynamics), the qualities may be turned to - using a consistency law between the connected nodes, derivated from the AI minimax algorithm. This may happen when a new arc is discovered. So, a CbMU may learn (modifies its quality values) only when a new feature in the dynamics in discovered.

Main hypothesis: the cycles within the perceptive graph are of special interest Remember that we want to fulfil a constraint at each time. For the perceptive graph possesses a finite number of nodes, some cycles may appear. Our hypothesis is that the cycles may be used to build the internal contexts of the CbMU.
Let's take the example of the pole-balancing problem with a 1-dimensional input space generated by θ. The constraint is $\theta \in [-0.2rad, 0.2rad]$ and the two basic tasks are "push on the left" and "push on the right".[-0.2,0.2] is divided into 10 states $S_1..S_{10}$ (fig. 3). The problem is clearly non-Markovian because we do not know the angular speed of the pole; we cannot build a successful policy if only one action is associated to each state.
Let consider a short term memory containing the last 5 states reached. If a state appears twice in this memory, a cycle has been performed and a new context is created (with its own policy). We can build a successful policy with the following rule:
(At the beginning of the trial, no special context) IF $\theta > 0$ PUSH ON THE RIGHT ELSE PUSH ON THE LEFT
(A cycle has been performed) Let $S = [S_{min}, S_{max}]$ be the last state in the short term memory. The policy is: IF $S_{max} > 0$ (IF $\theta > 0.04$ PUSH ON THE RIGHT ELSE PUSH ON THE LEFT) ELSE (IF $\theta > -0.04$ PUSH ON THE RIGHT ELSE PUSH ON THE LEFT)
Although this rule is very simple, it permits to balance the pole for 100000 steps at least, even with a 15 percent noise upon θ.
So, the basic idea is that when a cycle is discovered in a perceptive graph, a special node and a new meta-action are created: the special node means "I have just done this cycle" and the meta-action is

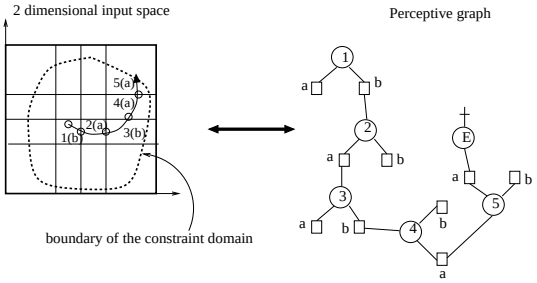

Figure 2: The perceptive graph of a CbMU.

the sequence of state/action performed in this cycle. And a context is the combination of the last cycle encountered and an area in the input space.

Advantages of the proposed method

- The hierarchical decomposition of the task through the different constraints permits to bring some a priori knowledge, reducing the input or the output space for each learning process. At each time, the decision involves different levels of contexts which filter the input data
- The CbMUs can cope with Partially observable Markov decision problems (POMDPs): the detection of cycles into the perceptive graph is used to build new internal contexts. A cycle is a kind of sub-goal which is memorised, like in the HQ-Learning method (Wiering and Schmidhuber, 1997). But the number of possible sub-goals does not need to be fixed at the beginning of the learning stage.
- A CbMU is able to adapt itself whenever a new arc is created in its perceptive graph, breaking the consistency law upon the qualities of some nodes.
- There are no internal parameters.
- The learning process is not CPU consuming, because it only consists on adding nodes or arcs and performing min or max operations upon the qualities of the nodes.

Drawbacks of the proposed method

- The learning process is designed for constraint based tasks (no optimal policy)
- The number of input signals must be small to have a reasonable number of nodes.

Figure 3: The pole-balancing problem with a 1-dimensional input space.

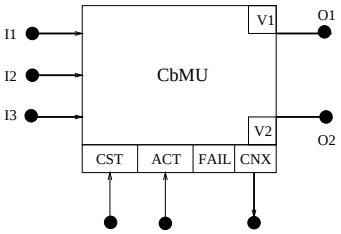

Figure 4: The external structure of a CbMU.

2.2 External structure of a CbMU.

A CbMU (fig. 4) is a black box composed of three kinds of inputs: the perceptive data, which is a vector $(I_1, ..., I_n) \in R^n$, the CST bit, which is the binary value of the constraint at time t, and the ACT bit, which is the current state of activation of the CbMU. An output channel among the vector $(O_1, ..., O_p) \in \{0,1\}^p$ may be fired only if the ACT bit is set to 1 (the CbMU is activated). At each time, one and only one channel may be activated. It represents the choice of the CbMU, given the perceptive data $(I_1, ..., I_n)$, in order to respect the constraint given by the CST bit.
The choice leads to a modification of the CbMU environment, so that it changes the values of the input data, leading to a possible change of the CST bit (fig. 5). The external available informations are:

- the binary qualities $(V_1, ..., V_p) \in \{0,1\}^p$ associated to the firing of the output channels. If V_k is set to 0, it means that the activation of the channel k is considered to lead (sooner or later) to a non-respect of the constraint CST (see paragraph 2.4).
- The FAIL bit, which is set to 1 if the learning procedure of the CbMU has failed (see paragraph 2.4)
- The CNX bit, which is set to 1 if the connexion to the CbMU is allowed (the ACT bit modification is permitted by the CbMU). The allowance condition is: FAIL=0 and CST=1. If the CNX bit equals 0, the ACT bit is automatically set to 0 (the CbMU disconnects itself).

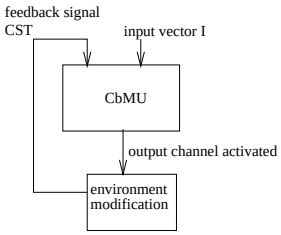

Figure 5: Diagram showing the links between a CbMU and its environment, while the CNX bit remains equal to 1.

2.3 Internal structure of a CbMU.

The CbMU is internally composed of two main items (fig. 6), which goal is to provide at each time a quality to the firing of each available output channel, given a particular input data:

- a set of perceptual areas $\{Z_1, ..., Z_p\}$, each one linked to a particular output channel. Each $Z_k \in R^{n_k \leq n}$ is connected to some of the input channels of the CbMU and is divided a priori into a set of b_k boxes $Box_k^{j, j \in \{1,...,b_k\}}$ created accordingly to the following set of equations:
$$\begin{cases} \forall k \in \{1, \ldots, p\} \bigcup_{j \in \{1,\ldots,b_k\}} Box_k^j = Z_k \\ \forall \{j, l\} \in \{1, \ldots, p\}^2, j \neq l, \\ Box_k^j \bigcap Box_k^l = \emptyset \\ Box_k^j = \{I = (I_1, \ldots, I_{n_k}) / \\ \quad \forall l \in \{1, \ldots, n_k\}, m_l^j \leq I_l < M_l^j \ \} \end{cases}$$
Thus, each box Box_k^j is parameterised by n_k couples of values (m_l^j, M_l^j) which are the boundary values for each perceptive input signal used by the perceptive area Z_k associated with the channel k of the CbMU.

- a set of pre-connected bits, whose initial value is 1, divided into two categories:

  1. the perceptual state bits P, each of them may be associated to a set of p boxes $\{Box_1^{j_1}, ..., Box_p^{j_p}\}$.
  2. the choice bits C, each of them pre-connected to a perceptual state bit.

  The pre-existing ending state E corresponds to a non-respect of the CST constraint (CST turns to 0).

The way the perceptual areas are divided is considered to be an a priori knowledge: it is not modified during the learning stage of the CbMU.

Short term memory and cycle detection It recalls the last 5 action nodes the CbMU has reached. If the last element of the short term memory is equal to one of the four others, a cycle has just been performed and the CbMU switches to a new context. All the contexts are associated to a precise cycle and possess their own perceptive graph; the system switches from a context K_i to a context K_j by performing the cycle associated to K_j in the perceptive graph of K_i.

2.4 Learning procedure of a CbMU.

Introduction The proposed learning 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 optimisation method, but on the respect of binary perceptive constraints. Moreover, each CbMU may learn (that is to say "adapts itself to correct a detected inconsistency between the real facts and the predicted ones") whenever it is activated.
The objective of each pre-connected set of bits is to evaluate the impact of a choice among the O_k on the evolution of the perception signal I received by the CbMU. The binary value of a P bit expresses the quality of the associated perceptive state, that is to say the capability of the CbMU to find a sequence of choices from this state in order to respect the constraint of the CbMU.The binary value of a C bit expresses the quality of a choice from a precise perceptive state.
The learning algorithm is based on two items:

- the on-line building of connexions between the sets of pre-connected bits(so called the perceptive graph), making an internal representation of the dynamics of the system.

- a consistency law between two connected bits of the perceptive graph, derived from the AI minimax algorithm (Rich, 1983).

The perceptive graph. The objective is to evaluate the impact of a choice among the O_k on the evolution of the perception signal I received by the CbMU. To do so, while the CST bit remains equal to 1, the CbMU possesses at each time a single active perceptual state P. When a failure is detected (CST turns to 0), the CbMU is in the special state E.
Thus, the CbMU A_i has a finite number of states, including an ending state E. The dynamics of the system makes the agent move from one state P to another state P', according to the choice C_i made

among the elements of O. The transition $P \rightarrow P'$ produces an arc between the pre-connected C_i bit of the perceptual state P and the perceptual bit P'. If the result of the choice O_i is a failure (the CST bit turns to 0), an arc is made between the pre-connected C_i bit of the perceptual state P and the ending state E (the E bit is always equal to 0).

The consistency law. The problem which consists on taking a decision according to the fulfilment of constraints, with a given dynamics of the system may be seen as a two players game: the CbMU and the dynamics. The aim of the CbMU is to never reach the losing state E , which quality q_E 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 value of the bits). The consistency law applied for each node (these are the bits) of the graph is given by the following two relationships. The first one is dealing about the perceptive bits P, whereas the second one is applied to the pre-connected bits C associated to a decision-making when the perceptive state of the CbMU is P:

$$q_P = \max_{C_i \in Child(P)} \{q_{C_i}\} \quad (1)$$

$$q_C = \min_{P_i \in Child(C)} \{q_{P_i}\} \quad (2)$$

Where $Child(P)$ is the set of the children of P in the graph and $Child(C)$ is the set of the children of C.

Using the consistency law to learn. As soon as an arc from a C_i bit to a P' bit is created while the dynamics makes the perceptual data of the agent evolve, $Child(C_i)$ may be modified, therefore the consistency relationship could be broken (fig. 7). In that case, the value of C_i is forced in order to respect the equation (2). If q_{C_i} is modified, the value of the quality associated with the father P 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.
If all the perceptual states are considered to lead to the losing state E, the learning stage has failed. Then, the FAIL bit (see fig. 4) is set to 1.

2.5 Decision-making.

The decision-making (that is to say the firing of an output channel) of an activated CbMU remains

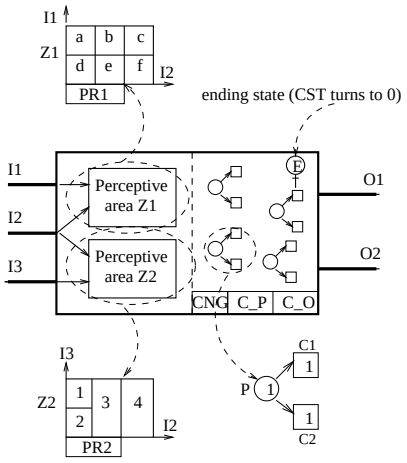

Figure 6: The internal structure of a CbMU: a set of two perceptive areas (Z_1 and Z_2 linked with the two output channels O_1 and O_2) and a set of pre-connected bits (P is a bit linked with a perceptive state of the CbMU = $(Box \in Z_1, Box \in Z_2)$), C_1 (resp.C_2) is a bit associated with the firing of the channel 1 (resp. 2), given that the perceptive state at time t is P. C_P is a pointer to the current activated P, C_O is a pointer to the current activated output channel and the CNG bit is set to 1 when the C_P bit has just been modified (transition between two perceptual states).

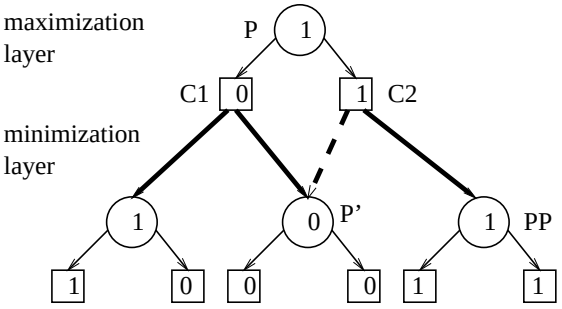

Figure 7: Detail of the graph associated with the learning process of a CbMU. The current perceptive state is P. The past experience of the agent allows it to detect the transition to one perceptive state (PP) from P when the choice C_2 is made. The activation of the output O_2 make the perceptive state turn to P', which creates a new arc (dashed line). Then, the consistency law is broken so that C_2 moves to 0 and, by retro-propagation, P turns to 0 too.

unchanged while the C_P bit (see fig. 6) is equal to 0 (the input data remains in the C_P state).
When a transition between two perceptual states is detected (the CNG bit turns to 1), the new fired output channel C_O is the one which associated V_{C_O} is maximal. If two or more channels cannot be discriminated with the former rule, the one which priority PR (see fig. 6) is maximal is elected. If two or more channels cannot be discriminated with this rule, one is randomly chosen.

2.6 Global learning algorithm.

A CbMU can be seen as an independent process which interacts with other processes, by the way of the ACT bit, the CST BIT, the CNX bit and the output channels O. Its learning phase begins as soon as the ACT bit turns to 1 (the CbMU is called by another process), given that the connexion is allowed by the CbMU (the CNX bit is equal to 1). It stops when the ACT bit turns to 0 (the calling process disconnect the CbMU) or the CST bit turns to 0 (the constraint is not fulfilled).
Step 1 - (when the ACT bit turns to 1) Retrieval of the internal parameters C_P and C_O (by the decision-making rule: see paragraph 2.5), given the current input vector I. The CNG bit is set to 0.
Step 2 - (while the CNG bit remains equal to 0) The output channel linked with C_O is fired.
Step 3 - [learning phase] (when the CNG bit turns to 1 and the CST bit is equal to 1) Retrieval of the new internal parameters C_P' and C_O'. If the bits pointed by C_O and C_P' are not connected yet, connect them and run the consistency law to eventually correct the value of the bits into the perceptual graph. Set the CNG bit to 0, and return to step 2.
Step 3' - [learning phase, failure detected] (when the CNG bit turns to 1 and the CST bit is equal to 0) If the bit pointed by C_O and the E bit are not connected yet, connect them and run the consistency law to eventually correct the value of the bits into the perceptual graph. Set the CNX bit to 0 (the CbMU disconnects itself).

2.7 Hierarchical connexion of CbMUs.

In this paragraph, a basic idea of how to connect CbMUs is given. The main objective is to divide the whole behaviour into smaller ones that are driven by particular constraints.
In the figure 8, the cart-pole problem is divided into

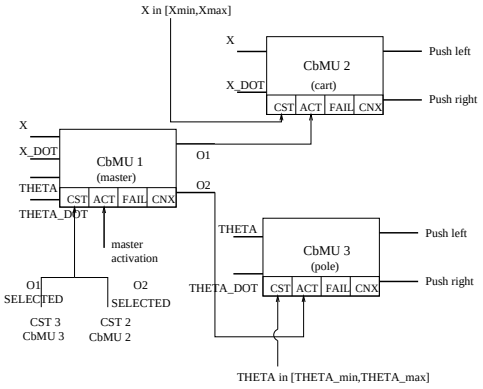

Figure 8: The cart-pole problem is carried out with three CbMUs: one is dedicated to the balance of the pole (CbMU 3), one is dedicated to the balance of the cart (CbMU 2) and one (the master) is coordinating the two other behaviours (CbMU 1).

three tasks, which are learnt together:

1. balance the pole (CbMU 3, CST 3)
2. balance the cart (CbMU 2, CST 2)
3. coordinate CbMU 2 and CbMU 3 (CbMU 1, CST1=CST3 if O2 is selected or CST1=CST2 if O1 is selected)

CbMU 2 and CbMU 3 are "specialised" and they are learning independently from each other, whereas CbMU 1 has to learn how to switch from a cart balancing context to a pole balancing context and vice versa. So, the ACT bit of CbMU 2 and CbMU 3 are connected to the output channels of CbMU 1: when CbMU 1 choose to fire O_1 (resp. O_2), CbMU 2 (resp. CbMU 3) is activated and choose among the "push left" and the "push right" task.
If CbMU 2 is chosen by CbMU 1 and CST 3 is not fulfilled, CbMU 1 has taken a mismatched decision (the control should have been given to CbMU 3 to balance the pole).

3 Validation experiments.

3.1 Context of the experiments.

The experiments have been carried out with the help of the Khepera simulator. Khepera (fig. 9) 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

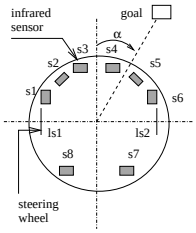

Figure 9: The miniature mobile robot Khepera.

Tasks	meaning	ls_1	ls_2
T_1	Move forward	3	3
T_2	Move to the right	2	0
T_3	Move to the left	0	2
T_4	Turn on the right	2	-2
T_5	Turn on the left	-2	2

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

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 behaviour, 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 is performed by a wall following behaviour, divided into two sub-tasks: follow the wall on the left and follow the wall on the right.
Four different input signals are utilised: $I = (d_{left}, d_{forward}, d_{right}, \alpha)$. 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 tasks have been chosen, which are linked to couples (ls_1, ls_2): $T = \{T_1, T_2, T_3, T_4, T_5\}$. Their specification is given by table 1.
The robot possesses three internal binary feedback signals, which are the three constraints being used by the CbMUs: BUMP, FWL and FWR. 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 than 10 for more than 30 learning steps.

3.2 High level goal-seeking algorithm.

The goal-seeking strategy followed by the robot is a high-level algorithm in which three contexts are considered: "reach the goal", "follow the wall on the left" and "follow the wall on the right".

[reach the goal]
If the goal is behind the robot and it can go forward, T_1 is executed.
If the robot is near from an obstacle and the goal is on the same direction, it switches to a context [follow the wall]

[follow the wall on the right (resp.left)]
If (the goal is on the left (resp. right) side of the robot and it can go on the left (resp. right) without colliding) or (the goal is behind the robot and it can go forward without colliding), it switches its context to [reach the goal].
Else the CbMU associated to a "follow the wall on the right (resp. left)" behaviour is utilised.

3.3 Specification of the CbMUs.

The algorithm described in the last paragraph shows that the strategy of the robot uses its perceptive data. The learning process focuses on their management. Both the hierarchical set of CbMUs and their internal constraints, using BUMP, FWR and FWL, are given by fig. 10. There are two master CbMUs A_1 and A'_1, which respectively accomplish the "follow the wall on the right" and "follow the wall on the left" tasks. A_2 and A'_2 CbMUs must avoid obstacles respectively by moving on the left and by moving on the right. 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.

The quality of the box fired in the perceptive area of the agent A_2 associated to the choice of T_3 is used by the robot to know if it can turn on the left without colliding (see paragraph 3.2). In the same way, the quality of the box fired in the perceptive area of the agent A'_2 associated to the choice of T_2 is used by the robot to know if it can turn on the right without colliding.

3.4 Learning protocol.

A learning process, which consists of trials/failures steps, is developed in the environment given by fig.

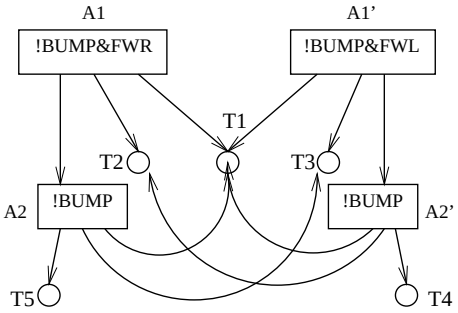

Figure 10: Hierarchical connexion of the CbMUs used by Khepera.

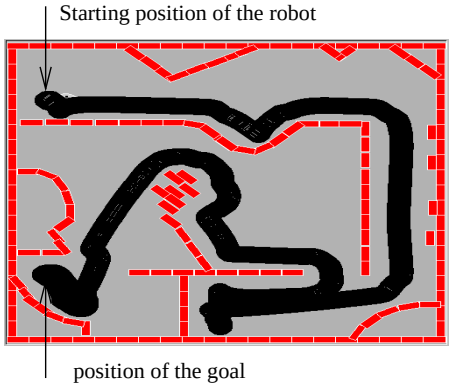

Figure 11: Goal-reaching behaviour in the Khepera simulator environment.

11. A trial ends when a failure is detected by one of the CbMUs or when the goal is reached.
All the CbMUs are learning from scratch together. In the next paragraph, we will consider that the learning stage is successful if none of the four implied CbMUs have made a mistake during 500000 consecutive learning steps.

3.5 Results.

10 learning attempts have been done for the global learning of the four CbMUs. All the attempts were successful, ending after 57 up to 258 trials. Fig.13 shows the evolution of the number of consecutive learning steps without failure (including all the CbMUs) for one of the attempts. It is noticed that the duration of a trial is a function of the number of nodes in the perceptual graph (fig. 14 is given for the CbMU A_2). It simply means that as soon as a CbMU has a wide perceptive experience, it is able to respect its constraints with making very few mistakes. According to the perceptive areas division process (see paragraph 3.3), the maximal number of nodes for all the CbMUs is $64+3\times64+1=257$. This number is nearly reached at the end of the learning stage. If the learning environment is changed to another one (fig. 12), the Khepera robot is able to go to the goal without any failure after only 15 trials (new perceptive data have been tested).

4 Discussion and future work.

The CbMUs are designed to permit an incremental learning of a global behaviour. The hierarchical connexion of the CbMUs allows to divide the whole behaviour into smaller ones, each of them driven by a particular perceptive constraint. According to its perception, the aim of each of a CbMU is to respect its internal binary constraints. To do so, it must decide to execute a basic task or to call another CbMU, to let it choose by itself.
The learning procedure of a CbMU is made by a low CPU cost algorithm which is based on the respect of an internal consistency law between the nodes of the built-on-line perceptive graph. The binary value of these nodes may be modified when a new arc is created into the graph. Using a simulated Khepera robot which aim is to learn a safe goal-reaching behaviour, it has been shown that the hierarchical set of CbMUs created for this task are able to learn at any time they are called. Thus, they can learn together in a high level algorithm framework where they have to face new perceptive situations and must adapt themselves whenever an inconsistency between what has happened in real and what was predicted is discovered. Besides, the algorithm can cope with very noisy perceptive data produced by the infra-red sensors of Khepera.
The quality associated to each node of the perceptive graph of a CbMU can be used to statically recognise some perceptive situations. Moreover, it should be possible to use the perceptive graph in a dynamic environment recognising process: the execution of the wall-following task generates a particular cyclic sequence of perceptive states for the wall-following agent. So, we are thinking of adding new kind of nodes associated to cycles into the graph to cope with dynamic recognition of situations.

References

Asada, M., Noda, S., and Hosoda, K. (1996). Action-based sensor space categorization for robot learning. In *IROS'96*, pages 1502–1509.

Bersini, H. and Gorrini, V. (1996). Three con-

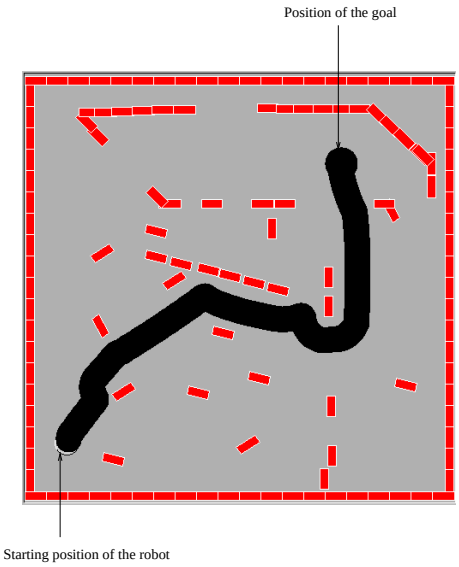

Figure 12: Goal-reaching in a new environment.

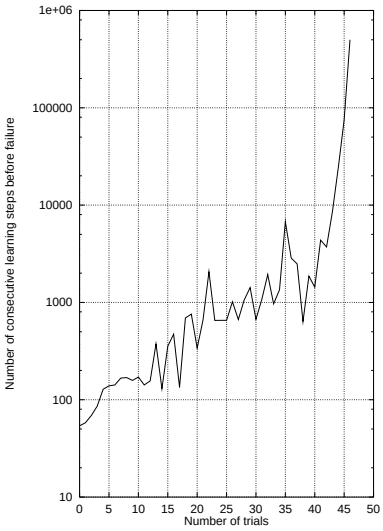

Figure 13: Evolution of the number of consecutive learning steps without a failure (including all the CbMUs).

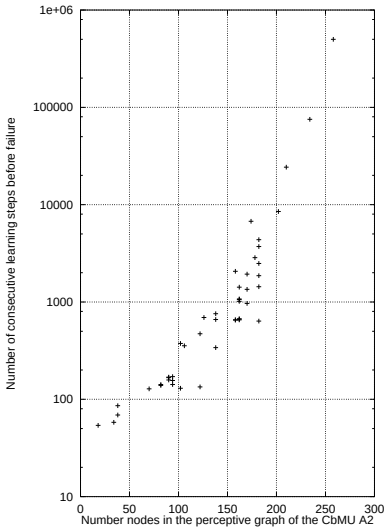

Figure 14: Relationship between the number of nodes within the perceptive graph of A_2 and the number of consecutive learning steps without a failure.

nectionist implementations of dynamic programming for optimal control: A preliminary comparative analysis. In *Workshop on Neural Networks for Identification and Control in Robotics*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321.

Michel, O. (1996). Khepera simulator package version 2.0: Freeware mobile robot simulator. `http://wwwi3s.unice.fr/~om/khep-sim.html`.

Mondada, F., Franzi, E., and Ienne, P. (1994). Mobile robot miniaturization: A tool for investigation in control algorithms. In Yoshikawa, T. and Miyazaki, F., editors, *Proceedings of the Third International Symposium on Experimental Robotics 1993*, pages 501–513. Springer Verlag,.

Rich, E. (1983). *Artificial Intelligence*. McGraw-Hill.

Watkins, C. (1989). *Learning from delayed rewards*. PhD thesis, Universit de Cambridge.

Wiering, M. and Schmidhuber, J. (1997). Hq-learning. In *Adaptive Behavior*, volume 6:2, pages 219–246.