Learning Plan Applicability through Active Mental Entities

Pietro Baroni, Daniela Fogli, Giovanni Guida

Università di Brescia, Dipartimento di Elettronica per l'Automazione
Via Branze 38, 25123 Brescia, Italy - fax: +39-30380014
e-mail: {baroni, guida, fogli}@ing.unibs.it

Abstract. This paper aims at laying down the foundations of a new approach to learning in autonomous mobile robots. It is based on the assumption that robots can be provided with built-in action plans and with mechanisms to modify and improve such plans. This requires that robots are equipped with some form of high-level reasoning capabilities. Therefore, the proposed learning technique is embedded in a novel distributed control architecture featuring an explicit model of robot's cognitive activity. In particular, cognitive activity is obtained by the interaction of *active mental entities*, such as intentions, persuasions and expectations. Learning capabilities are implemented starting from the interaction of such mental entities. The proposal is illustrated through an example concerning a robot in charge of reaching a target in an unknown environment cluttered with obstacles.

Keywords: Intelligent Agents, Mental activity, Autonomous Robots, Learning, Plan Modification

INTRODUCTION

The issue of endowing autonomous mobile robots (AMRs) with learning capabilities has received a great deal of attention in recent years. In fact, learning capabilities are required to enable a robot to adapt to changes in the external world and to develop new operation strategies in front of unknown situations. In other words, learning capabilities are necessary to achieve real autonomy in a dynamic and unpredictable world.

Robots designed according to a "no learning" approach, that is without embedded learning mechanisms, need a built-in endowment of knowledge and capabilities that can guarantee the ability to deal with unforeseen situations the robot may meet during its operation. Such a built-in endowment is feasible only in controlled environments, where a limited degree of autonomy is required to the robot and no unpredictable situations may occur.

Learning without knowledge

The failure of the "no learning" approach led to the development of a dual approach, where it is assumed that the robot has no (or very limited) a priori knowledge but is endowed with learning mechanisms which allow it to progressively improve its capabilities. Let us call this approach the "no knowledge" approach.

A typical experimental context according to the "no knowledge" approach can be described as follows. The robot is located in a static environment and has to accomplish one (and only one) simple task. Initially, the robot has no a priori knowledge about the environment nor about the way the task can be actually performed. After the learning activity, it is able to perform the stated task in the same environment in a more efficient way than it was able in the initial state.

Neural Networks, Learning Classifier Systems and Evolutionary Computation are among the most important learning techniques actually adopted in this kind of learning experiments with autonomous mobile robots (8). The learning paradigms underlying such techniques, such as Reinforcement Learning (5) or Evolutionary Learning (11), share the same basic idea: a cost (or performance) function is defined in correspondence of the task to be accomplished and has to be optimized in order to realize the task. For instance, in neural networks, reinforcement learning "aims at maximizing the expected evaluation by adjusting the parameters of the network" (12). Evolutionary algorithms can be used in a

similar way to evolve neural controllers (9) (1). A learning classifier system uses, in general, a reinforcement learning algorithm to assign rewards or penalties to the rules of the system (6) (7) and, in some cases (as in (6)), a genetic algorithm is exploited to overwrite weak rules with new more powerful ones.

As a typical example of application of such learning techniques, many authors have considered the problem of learning how to reach a target in an unknown environment cluttered with obstacles (6) (7) (14). In this case, the robot receives a penalty every time it collides with an obstacle, while it receives a reward when it gets nearer the target. In practice, the cost function says "good actions allow robot to approach the goal, whilst bad ones cause collisions with obstacles". This crucial knowledge is hardwired into the definition of the cost function and of the penalty/reward mechanism. Therefore, the learning procedure is conceptually rather simple: first, an action is chosen randomly and actually performed; then, the new value of the cost function is evaluated and it is compared with the value before the action; finally, a penalty or reward is calculated and associated to the performed action, in order to assess its eligibility in a similar situation.

Although this approach to learning has given encouraging results in the sort of experimental contexts described above, it relies on some critical assumptions, whose validity in more realistic contexts is questionable.

First of all, it is assumed that the robot is capable, after every action, of evaluating if that action has brought it nearer the goal. For example, it is hypothesized that the robot can always know its position in order to determine the suitability of its movements towards the target (7); but, in general, this is not the case in practice. Delayed reinforcement learning solves this problem by rewarding the robot only when it reaches the target; but this method has the drawback of being so complex and so slowly to converge, that it is completely useless in real applications (6).

Moreover, it is assumed that the cost function and the penalty/reward mechanism cover all the cases the robot may meet, in the sense that in any situation, they provide a correct indication about how the robot should modify its behavior. However, in real world, many unpredictable situations have to be managed and it may not be reasonable to require that predefined functions are able to correctly deal with all possible cases.

Finally, usually only one (simple) task is considered: wall-following, light orientation, obstacle avoidance, locating food sources, box pushing, are among the most advanced behaviors actual robots can learn. However, in real-world applications, an autonomous robot has to deal with several complex tasks, which may need to be carried out in parallel and may conflict. Even if these learning techniques have given satisfactory results with simple tasks, their application to more articulated problems seems to be critical for many reasons. Just to mention one, the definition of the cost function and of the penalty/reward mechanism is not straightforward when facing a complex task. Also Dorigo (8) seems to share this perplexity: "given the relatively young age of the field, the applications [...] are relatively simple when compared to what we would like our robots to be able to do", whereas Kröse (12) asserts, "for more complex applications these methods are inadequate".

Learning through cognitive activity

In our opinion, the assumptions made about the initial capabilities of the robot in the "no knowledge" approach are too strong and determine its inadequacy in solving complex problems. In fact, the "no knowledge" approach represents a sort of extreme reaction to the "no learning" approach and, as such, may in turn fail to be practically applicable outside some carefully designed contexts. It is undoubted that built-in knowledge alone can not be sufficient for realizing fully autonomous robots. However, this does not necessarily imply that every built-in knowledge should be suppressed, every explicit knowledge representation should be avoided, and everything should be learned from scratch using implicit subsymbolic representations.

In fact, instead of suppressing built-in action plans, one should consider the opportunity of modifying and improving them, thanks to learning activity. In other words, some programmer-provided knowledge is useful to face complex tasks; however, the robot should be able to not blindly trust and apply it, but to criticize and improve such knowledge, when it proves to be inadequate. This requires that the robot is equipped with some form of high-level reasoning and anticipatory capabilities: the robot should be able to anticipate its course of action on a sufficiently large time horizon, an then to compare the result of its anticipatory activity with the actual results of actions then carried out in the world. The outcome of such comparison is the basis of learning activity.

This kind of capabilities is normally not considered in the "no knowledge" approaches. In fact, in such kind of approaches, no reasoning activity underlies robot actions, which are merely reactive. The adopted learning methods allow only to establish which action is good and which one is not. The reasons making an action good or not are neither represented nor learned.

In our view, the robot should be provided with some built-in knowledge, provided that this knowledge is not regarded as infallible. The robot may exploit this knowledge in order to carry out its tasks, but simultaneously, it has to be able to

recognize the limits of its own knowledge and, if necessary, to refine and improve it.

This requires that the robot has some sort of awareness of the fact that, when it performs an action, it is devoted to the achievement of some goal. Therefore, if the goal is not reached, the robot may wonder if the action was really suitable to reach such goal or if there is some external condition that makes the action unsuitable in the current situation. This may then lead to a revision and refinement of robot knowledge.

This alternative approach to learning requires therefore that the robot has an explicit representation of the goals it is pursuing, of the association between a goal and the action plan that is intended to achieve it, and that it is able to detect when the plan fails to reach the goal. In other words, this approach requires that the robot takes its actions mainly on the basis of expectations of future world states, derived through its knowlegde, i.e. that the robot acts as an anticipatory system (18).

Interestingly enough, such requirements are shared by a consolidated research trend which suggests that an explicit representation of mental entities (such as beliefs, desires, intentions, etc.) is required to provide autonomous systems with intelligent behavior (10)(17)(16)(19). Such works have stressed the importance of explicitly modeling robot cognitive activity in order to achieve high-level capabilities, but the issue of exploiting cognitive activity for learning has not been deeply explored yet.

Therefore, the aim of this paper is to carry out a preliminary investigation about how learning capabilities could be introduced in a control architecture featuring an explicit model of robot cognitive activity.

As a background we assume the control architecture based on the concept of active mental entities introduced in (2)(3)(4) and we show how learning capabilities can naturally and effectively be realized by exploiting the peculiarities of such kind of control architecture.

The rest of the paper is organized as follows. Section 2 is devoted to the description of the structure and operation of the architecture. Section 3 explains our learning technique based on plan failure recognition and plan refinement. Section 4 illustrates how the learning technique can be applied to an application example. In section 5 the peculiarity and the advantages of our proposal are summarized.

A CONTROL ARCHITECTURE FOR AMRS BASED ON EXPLICIT REPRESENTATION OF MENTAL ENTITIES

In this section we briefly survey a distributed control architecture for AMRs, based on the explicit representation of robot mental entities.

The basic ideas underlying our approach can be summarized as follows:

i) the mental processes occurring inside an autonomous agent, like a mobile robot are the result of the cooperation and conflict between a set of mental attitudes;
ii) since the interactions between attitudes are expected to produce a globally intelligent behavior, it is essential that mental attitudes are provided with individual and independent operation capabilities;
iii) as a consequence of (i) and (ii), we represent mental attitudes as *active mental entities*, i.e. entities that can freely interact in a distributed context according to their own nature;
iv) active mental entities give existence to dynamic processes that can be created and disposed when necessary.

First, we define intentions, persuasions and expectations, the active mental entities which represent the basic elements of our model of cognitive activity. Then, we illustrate the structure and the operation of the resulting robot control architecture.

Intentions

An *intention* can be conceived as the will of reaching a particular state of the world. That is, an intention expresses the concept: “I want to pursue something”. Thus, we assume that an intention is an autonomous active entity definitely committed to reach its achievement, called the *subject* of the intention. Sometimes, pursuing a particular intention can be impossible or inconvenient or meaningless; for this reason, we assume that intentions must rely on some *validity conditions*. For instance, the intention “I want to find Mr. Smith” is valid for the robot only under the condition that it believes that it is possible for it to find Mr. Smith and, of course, Mr. Smith has not been found yet. We distinguish between *generated intentions*, which are created as a consequence of the interaction between other mental entities, and *primitive intentions*, which are always active inside the robot (such as “I want to preserve my physical integrity”) and which do not rely on any validity condition.

To achieve its subject, an intention has to select and carry out a *plan*. A plan is, in the most simple case, a sequence of tasks. A *task* is a piece of work to be undertaken and can be represented by a simple primitive action (a computation, a sensorial acquisition, an action on the environment) or by a more complex and less definite action whose purpose is to achieve a particular state of the world. The latter case obviously leads to the creation of a new intention for the accomplishment of the task.

We assume that an intention is able to generate plans; this can be performed by using a suitable mechanism in charge of plan generation or by searching in a collection of precompiled plans. Each plan has an associated *applicability condition*, characterizing the situations where it can be applied. The intention, according to the applicability conditions, selects the most suitable plan for the current situation, and then puts it at work. As it will be better explained later, each intention is in relation with one or more persuasions which are in charge of monitoring external conditions. If a relevant change in the external world is detected, this is notified to the intention which has then to revise its plan accordingly.

In more precise terms, we can define an *intention* as a four-tuple I = <S, C, P, IM>, where:

- S is the *subject* of the intention, namely a proposition describing a desired state of the world or a goal;
- C is the *validity condition* that enables the subject S to be a valid goal;
- P is the current *action plan* that is supposed to achieve the subject S under the condition C;
- IM is a set of *methods* used by the intention in its operation.

Initially, when an intention is generated, only S, C and IM are given; P has to be constructed, executed and then dynamically revised by the intention itself through its operation. Therefore, the set of methods IM must include all the procedural elements necessary for the intention to effectively carry out these tasks.

The set of methods IM is in turn a five-tuple IM = <Gm, Sm, Em, Rm, Cm> where:

- Gm is a *generation method*, i.e. a method for generating candidate action plans capable of achieving the subject S under the condition C;
- Sm is a *selection method*, i.e. a method for the selection of one of a candidate plan P to exploit for the achievement of the subject S under the condition C;
- Em is an *execution method*, i.e. a method for the execution of the selected plan P;
- Rm is a *revision method*, i.e. a method for the revision of the currently active plan P when changes occur that affect the validity condition of the currently active plan or of other candidate plans;
- Cm is a *conflict resolution method*, i.e. a method for solving possible conflicts between intentions.

Persuasions

A *persuasion* arises when it is of interest for the robot to know the *truth value* of a given *proposition*, that expresses a fact, an event or a property of the world. For this reason, a persuasion is generated when a new interesting proposition is met and is dismissed when the interest in the proposition ceases. Therefore, a persuasion is a persistent entity, that remains active until the proposition to which it refers remains interesting. A persuasion is thus conceived as an autonomous active entity, definitely committed to find and verify elements that can support the belief or disbelief in a proposition.

The activity of a persuasion is carried out by a process of searching *justifications* about the truth value of the related proposition. The justification for a truth value may be based on long-term knowledge, on sensorial data or, in turn, on other persuasions. Therefore, a persuasion may update or revise the truth value of the associated proposition when long term knowledge changes, new sensory data are acquired or other persuasions have revised the truth value of their own related propositions.

In more precise terms, we can define a *persuasion* as a four-tuple P = <S, V, J, PM>, where:

- S is the *subject* of the persuasion, namely a proposition whose truth or falsity is of interest for the agent;
- V is the currently most believed *truth value* of the subject S;
- J is the current *justification* that supports the assignment of the truth value V to the subject S;
- PM is a set of *methods* used by the persuasion in its operation.

Initially, when a persuasion is generated, only S and PM are given; V and J have to be constructed and then dynamically revised by the persuasion itself through its operation. Therefore, the set of methods PM must include all the procedural elements necessary for the persuasion to effectively carry out these tasks.

The set of methods PM can be represented as a five-tuple PM = <Gm, Sm, Vm, Rm, Cm> where:

- Gm is a *generation method*, i.e. a method for generating candidate activities for justification building, i.e. activities whose execution can provide evidence for the association of a truth value V to the subject S;
- Sm is a *selection method*, i.e. a method for the selection of candidate activities to exploit for building the current justification, i.e. for the determination of the current truth value V to associate to the subject S;
- Vm is a *verification method*, i.e. a method for the verification of the selected candidate activities, i.e. for determining whether one of the selected activities can actually lead to the construction of a justification and to the assignment of a truth value V to the subject S;
- Rm is a *revision method*, i.e. a method for the revision of the currently believed truth value V of the subject S when changes occur that affect the justification currently adopted for supporting the assignment of V to S;
- Cm is a *conflict resolution method*, i.e. a method for solving possible conflicts between persuasions.

Expectations

When a plan for the achievement of an intention has been completed, an *expectation* is created, representing the fact that it is expected that the execution of the plan has lead to the achievement of the intention. The expectation has a *subject* which is a proposition asserting the achievement of the intention subject, and a *truth value* which is true by default, since there exists the implicit assumption that, after plan execution, the subject of the intention is achieved. In order to verify whether an expectation is actually well founded, it refers to a persuasion with the same subject. The expectation may then be confirmed or refuted according to the truth value of this persuasion. For instance, some sensorial evidence could suggest that the subject of the intention has not actually been achieved. In this case a conflict arises between the expectation and the persuasion, which can be solved by hypothesizing that the executed plan failed in accomplishing the intention.

This mechanism is at the basis of our learning technique. When the failure of a plan is recognized, an investigation about the reason why the plan failed is carried out. At the end of this investigation, some actions are performed in order to learn a new plan. We will explain these points in more detail in section 3.

More precisely, we can define an *expectation* as a pair E = <S, EM> where:

- S is the *subject* of the expectation, namely a proposition representing the achievement of an intention subject;
- EM is a set of *methods* used by the expectation in its operation.

The set of methods M is in turn a pair EM = <Vm, Fm> where:

- Vm is a *verification method*, namely a mechanism for verifying if the persuasion having the same subject of the expectation has determined a truth value true;
- Fm is a *plan failure detection method*, that is a mechanism for notifying the interested intention that a failure occurred for one of its associated plans.

Relationships among mental entities

The following relationships are defined among the mental entities defined above.

When an intention is active, its subject and its validity condition become interesting propositions: therefore, relevant persuasions are created. Persuasions are involved also in the phase of plan generation on behalf of an intention, since also the applicability conditions of the alternative plans to be selected become interesting propositions.

During the process of searching justifications for the truth value of the related proposition, a persuasion may require the acquisition of sensorial data from the environment; this implies the creation of new intentions carrying out the data acquisition task. In this way, a continuous intention-persuasion interaction exist. Intentions generate persuasions which may generate or suppress other intentions, which may generate other persuasions, and so on.

An expectation is generated when the execution of a plan associated to an intention is completed. The subject of the expectation coincides with the subject of the relevant intention, so a persuasion with this subject is already active. The correspondence between the expectation and the persuasion is then exploited in order to verify if the intention has actually been achieved.

The overall architecture and its operation

We summarize here the main aspects of an AMR control architecture based on the active mental entities introduced above. For the sake of brevity and clarity the description is quite simplified. A more complete and detailed illustration can be found in (3).

The proposed architecture is made up of a collection of *components* that can communicate and cooperate in order to provide the robot with a global intelligent problem-solving behavior.

Components are classified into two types:

- *operative components*, in charge of performing actions, either physical, concerning the interaction with external world through sensors and actuators, or symbolic, such as computational and reasoning activities;
- *mental components*, that is intentions, persuasions and expectations.

All components are understood as active interacting entities. Mental components interact among them and activate operative components which, besides performing actions, may provide a feedback to mental components about the results of the actions carried out. It is assumed that all interactions among components are carried out according to a message-passing paradigm.

It is worth to remark here that the resulting architecture has the basic features of a Computing Anticipatory System. In fact, the operation of intentions, based on the achievement of goals through action plans, involves the existence of an anticipatory mechanism, driving plan selection and execution. This mechanism is made explicit by the presence of expectations, which show the crucial importance of the anticipatory perspective also in learning activity.

It should also be noted that, since a form of anticipatory behavior is ascribed to individual active mental entities, our conception of the internal structure of an agent is rather peculiar. In fact, in the context of the active mental entities paradigm, thanks to the nature of mental attitudes and to their capability of autonomously interacting one another, an agent can be viewed not just as an anticipatory system, but as a community of "small" anticipatory systems. This perspective, is very fruitful of future developments: in particular, the study of the properties of the global behavior emerging from an "eco-system" of interacting anticipatory entities is, in our opinion, particularly interesting.

The overall operation of the proposed architecture results from the autonomous operation of its internal components. We assume that each component is endowed with individual resources and that it can operate in parallel with the other components.

A dedicated component is in charge of interacting with the user and of receiving its requests. If the request is simple and an operative component able to satisfy it is available, the problem is directly addressed to it. Otherwise, if the incoming problem is more complex and can not be directly solved by a single operative component, a new intention is generated, whose subject coincides with the solution of the problem. The intention is in charge of identifying a set of alternative plans that may lead to the problem solution, of selecting one of them and of realizing it by resorting to the cooperation of other (mental or operative) components. While the firstly selected plan is executed, the intention continuously monitors the environment, through properly generated persuasions, and revises the current plan or even adopts a new one, if this is required by changes in the external situation. In fact, external events, such as environment changes or new requests from the users, do influence the operation of the system: new mental entities may be generated or some existing ones may be deactivated, current plans may be revised, and relationships between components may be modified, thus implementing a sophisticated reactive behavior.

After a plan is completely executed, the expectation that the intention subject is achieved is created. If it conflicts with current persuasions, plan failure is detected and the learning procedure presented in the next section is performed.

THE LEARNING TECHNIQUE

The proposed learning procedure starts when plan failure occurs. For the sake of simplicity, we will focus in this paper on the case where the reason of plan failure is an incomplete specification of the applicability condition of the plan. This means that there are some situations (unknown to the robot) where the plan does not work. The goal of the learning activity is to identify the correct conditions that make the plan applicable (or inapplicable) and to discover a new more suitable plan to be adopted in the situation where the old one fails. This activity includes the following three steps: plan failure detection, plan review, and plan consolidation. For the sake of brevity, only an informal sketch of

these steps is provided in the following of this section. The example in section 4 will show how the technique works in an application case.

Plan failure detection

As described above, plan failure is detected when a conflict arises between an expectation and the currently held persuasions. Such a conflict represents a symptom of *plan failure*, since, though the plan is terminated, there is evidence that the intention subject has not been achieved.

Some advantages of this mechanism for detecting plan failures are worth to be remarked. First of all, it is very general: it applies to every intention the robot pursues and does not depend on a programmer-defined reward or cost function, valid only for specific tasks or in particular contexts. In other words, every time the robot acts, it has an opportunity to learn: in a sense, learning is not confined to a limited initial phase but spans over the whole operation life of the robot.

Moreover, this method is only partially affected by the credit assignment problem related to delayed reward. In fact, plan failure may be detected for each intention adopted by the robot. Since an intention concerning a very complex task gives rise to the creation of other intentions concerning simpler subtasks, failures may be detected and possibly solved at a local level, rather than at the level of the initial intention. This represents a reasonable compromise between verifying the result only at the end of the execution of the global task and requiring that after each elementary action the robot is able to evaluate whether its effect has been positive or negative.

Finally, the interaction between intentions, expectations and persuasions allows the robot to be endowed with a sort of awareness. In fact, it is "aware" of which state of the world he wants to reach (the subject of the current intention). Moreover, it expects that this state is actually achieved after having executed a particular plan (an expectation is created). Finally, it is capable of recognizing whether its expectation is fulfilled or not (by exploiting the current persuasions about the real world).

Plan review

After plan failure is detected, the reasons underlying failure are investigated. In general, the fact that a plan has failed may be due to two causes:

The plan is wrong: it is doomed to always fail.

The plan sometimes fails: under some unspecified conditions the plan is unable to accomplish the intention to which it is associated.

In order to distinguish between these reasons, we assume that to each plan P a couple $Count(P)$ of numbers is associated. $Count(P)$ consists of two counters: $Nsucc$, the number of times P has been used successfully and $Nfail$, the number of times P has failed. In very general terms, cause (1) above is characterized by $Nsucc = 0$ and $Nfail > 0$, while cause (2) is characterized by $Nsucc > 0$ and $Nfail > 0$ (hopefully $Nsucc > Nfail$).

Let us suppose now that a failure has been detected for a plan which has been successfully adopted in the past. The robot has now to face two problems:

(a) identifying the reason of the failure;
(b) learning a new action plan which is able to achieve the current unsatisfied intention and that can be applied to similar situations in the future.

For problem (a) we consider in this paper only the case where the reason of the failure is represented by a missing condition which may appropriately restrict the applicability of the plan. In order to identify such condition, we adopt the heuristic of considering recent changes occurred in the environment as the most plausible clues of the existence of such a condition. No special mechanisms are needed to detect these changes, since they are easily identified by the persuasions whose purpose is monitoring the external world. So, if the truth value of some persuasion has recently changed, this is notified to the intention whose plan has failed.

Problem (b) requires the formulation of a new plan which allows the achievement of the intention in the current situation. After the new plan is formulated, it is executed and its ability to achieve the intention is verified. Plan generation and testing is repeated, if necessary, until the intention is finally achieved. The formulation of new plans to be tested could greatly profit from domain-dependent heuristics; however, we assume in this paper that a new plan is obtained by introducing a random modification within the previously applied one.

Once both problems (a) and (b) have been solved, it is possible to draw a relation between the subject of the persuasion(s) identified in the step (a) and the new successful plan obtained in step (b). In other words, an *applicability condition* is assigned to the new plan, which is added to the current collection of plans.

If no changes have been recently detected in the environment, the reason causing plan failure can not be identified in a simple way and problem (a) can not be solved. Phase (b) can be carried out anyway and a new plan can be formulated. In this situation, however, no applicability condition for the new plan can be identified. The new plan is stored together with the old one and considered as a possible alternative to it, though the reasons to select either one are not clear. This situation can be regarded, however, as a special case where the applicability condition of the new plan is coincident with that of the old one. Therefore, in the following we will consider explicitly only the case that a new applicability condition has been identified.

Plan consolidation

A consolidation phase is needed, in general, for evaluating if a stored new plan is really effective and reliable.

The consolidation phase is important to evaluate the results of the plan review phase, where a new plan *PNew* has been generated and an applicability condition has been associated to it. Of course, after one trial only, it is not sure that such an applicability condition is correct and that it is representative enough of the particular situation where the new plan should be selected. Only after the plan has been executed repeatedly, it is possible to reliably evaluate if *PNew* can be definitely considered correct or if there exist other exceptions to be taken into account.

During the consolidation phase, the old plan *POld* is still selected as the first choice. However, in order to compare *POld* with *PNew*, *Count(POld)* is not sufficient. In fact, *Count(POld)* refers to all cases where the old applicability condition is met. Nevertheless, *PNew* should substitute *POld*, only in the cases where the new, more specific, applicability condition holds. Therefore, a focused evaluation of the old plan, referring only to the new applicability condition, is necessary. For this reasons, another couple of counters *Count*(POld)* is created and updated only in the cases where the old plan is used in presence of the new applicability condition.

When the old plan fails and the applicability condition associated to *PNew* holds, this one is executed and its couple of counters *Count(PNew)* is in turn updated.

After the applicability condition of the new plan have been met a significant number of times, *Count*(POld)* gives an account of its actual applicability in such condition. On the other hand, *Count(PNew)* gives an account of how much *PNew* is suitable to deal with such particular condition. Thus, if *POld* almost always failed, whereas *PNew* almost always succeeded, the applicability condition of *POld* is modified, so that henceforth only *PNew* can be selected when its applicability condition is met. However, it can also happen that, at the end of the consolidation phase, the situation is not so clear: two alternative plans may appear to be successfully applicable in the same situation. In fact, it may occur that *POld* sometimes succeeds, even when the applicability condition of *PNew* is met. In this case there is no reason to modify the applicability condition of the old plan. However, the new plan represents a valid alternative when its own applicability condition is met and the old plan fails. In such a situation, there will coexist two applicable plans: *POld* with a looser applicability condition and *PNew* having a more restrictive condition. Therefore, the choice between plans becomes not so straightforward and if the failure rate of the old plan is limited, *POld* remains the first choice. Anyway, when *POld* fails, *PNew* represents a sort of spare solution, that can be directly attempted so skipping the plan review phase.

The following section illustrates an application of these learning methods to the case of a simple mobile robot.

AN APPLICATION EXAMPLE

In this section, we present an example of application of the previously illustrated learning technique which allows a robot to learn how to avoid obstacles and then, to learn how to deal with situations where both movable and immovable obstacles are present in the environment.

We hypothesize that our robot is capable of moving in four directions (north, east, south, west) over a square grid. The robot then includes infrared sensors to detect the proximity of objects in front of it, behind it, and to the right and the left sides. The only task that it can perform is reaching a particular target identified by a light which can be perceived by the infrared sensors. The described structure and capabilities are those ones of the miniature robot Khepera (15). We are currently implementing our control architecture and learning techniques on a Khepera simulator (13) which will be used as a first testbed for experimenting our ideas.

At the beginning, the request of reaching the target is accepted by the robot, and an intention whose subject is “go-to-the-target” is consequently generated.

We assume that the intention can exploit some built-in plans; for example, if the light is not seen, there could exist a plan for exploring the environment in order to search the light and then approach it. Available plans feature an applicability condition; so, the aim of learning is to modify them or their applicability condition (or both), in order to manage situations where they turn out to be unsuitable.

We will focus here on the following plan which is exploited by the intention when the target is already seen:

Plan1: Applicability Condition: light seen = "true"
task 1: turn towards the light
task 2: estimate the distance D from the light
task 3: go forward covering a distance D

Next, we will show how, after the failure of this plan in presence of obstacles, a learning activity starts with the purpose of developing an alternative strategy of action.

Learning to avoid immovable obstacles

Plan1 works well when no obstacles are present between the robot and the target.

Let us now suppose that, while the robot is moving towards the light, an immovable obstacle is met preventing robot’s movement, but not preventing it from perceiving the light. The plan keeps on being executed, so robot’s wheels keep on turning, but the robot remains still. After a number of wheel revolutions corresponding to distance D, the plan is completely executed and the expectation, whose subject is “target-reached”, is created. The truth value of this expectation is set to ‘true’ by default. Then, the expectation waits for a notification coming from the persuasion having the same subject. This persuasion notifies to the expectation that the target is not reached, since the light is not perceived at the maximum level of intensity. Plan failure is thus detected and a learning procedure starts. First of all, a new plan is created to deal with the current situation; it is obtained by adding random actions to the available plan. In particular, since the plan concerns movement, random movement actions are added. Moreover, they are added at the beginning, rather than at the end of the plan, since they are intended to modify the situation where the old plan failed and then to reapply it, rather than to reformulate it from scratch. For instance, the following new plan can be formulated in this case:

New Plan: Applicability Condition: light seen = "true"
task 1: turn in a random direction
task 2: go forward for a random time
task 3: turn towards the light
task 4: estimate the distance D from the light
task 5: go forward covering a distance D

This plan is put at work and trials with different random choices are repeated until the light is reached.

Simultaneously, reasons of plan failure are investigated. To do this, notifications about recent changes in the environment are requested to all persuasions in charge of monitoring the environment. In this case, the persuasion receiving data from the proximity sensors and whose subject is “object-in-front” notifies that its truth value has recently changed from 'false' to 'true'. Then, the subject “object-in-front” of the above persuasion is related to the new plan and becomes a part of its applicability condition. An alternative plan is thus added to the strategies of action associated to the intention “go-to-the-target”:

Plan2: Applicability condition : light seen = "true" & object in front = "true"
task 1: turn in a random direction
task 2: go forward for a random time
task 3: turn towards the light
task 4: estimate the distance D from the light
task 5: go forward covering a distance D

This one becomes a plan under consolidation, since its relation with the condition "object in front" needs to be verified. So, in next trials, *Plan2* is chosen only if *Plan1* fails. During these trials, the number of successes and failures of these plans is updated. In this example, since the relationship established between the plan and the applicability condition was correct, after some trials it emerges that *Plan1* fails every time an obstacle is met, whereas *Plan2* succeeds.

Thus, at the end of the consolidation phase, *Plan2* is definitely added to the collection of plans associated to the intention "go-to-the-target". On the other hand, since *Plan1* cannot be used in all situations where "light seen" is 'true' (in fact it always fails in presence of obstacles), its applicability condition is modified. Thus, *Plan1* becomes:

Plan1**: Applicability Condition:* light seen = "true" & object in front = "false"
task 1: turn towards the light
task 2: estimate the distance D from the light
task 3: go forward covering a distance D

In the future, *Plan2* will be chosen every time an obstacle is met on the path.

Learning to deal with movable and immovable obstacles

We consider now a more complex situations, where both movable and immovable objects are present in the environment, assuming again that the robot initially possesses only *Plan1* to reach the target.

When robot collides with an immovable obstacle, the situation illustrated in subsection 2.1 occurs again and *Plan2* becomes a new strategy under consolidation. However, during this consolidation phase, *Plan1* is always chosen first; so, when the robot collides with a movable obstacle, it keeps on going forward by pushing the object and by dragging it towards the target position. In this case, *Plan1* succeeds in a situation where the applicability condition of *Plan2* holds. Therefore, at the end of the consolidation phase, the couple of counters *Count*(Plan1)* records a certain number of successes and failures.

In particular, if several movable obstacles have been met during the experiments, the score of *Plan1* remains pretty good; so it remains unchanged. On the other hand, *Plan2* has showed to be sometimes successful, if at least some immovable obstacles have been encountered. Thus, since both plans result applicable when the light is perceived by the robot, the choice between them will depend on the current value of *Count*(Plan1)* and *Count(Plan2)*.

Therefore, the following behavior emerges in the long term: *Plan1* is anyway executed when an obstacle is met, thus movable obstacles are always pushed by the robot until the target. If an immovable obstacle is on the path, *Plan1* fails. Then, since there exists an alternative plan which can be applied (*Plan2*), this one can be attempted first, instead of immediately starting a new learning phase.

On the other hand, a different behavior emerges if the robot meets few movable objects and many immovable obstacles. This case is treated as that one presented in subsection 2.1; that is, *Plan1* is substituted by *Plan1** having a more restrictive applicability condition. Both movable and immovable obstacles will be always avoided by exploiting *Plan2*, since the presence of an obstacle makes false the applicability condition of *Plan1**.

CONCLUSIONS

In this paper we have presented a preliminary investigation of a novel approach to learning for autonomous mobile robots. Two basic assumptions underly it:

- some a priori knowledge can be provided to robots in order to enable them to face complex and unpredictable situations;
- cognitive activity can help in incrementally refining and improving this knowledge.

The first assumption arises from the consideration that current learning techniques used in robotics seem too weak and limited for being applied in complex contexts. In our opinion, this is due to the fact that the goal of learning everything from scratch using simple mechanisms, such as the penalty/reward one, is too ambitious. On the other hand, if robots are initially provided with some built-in action plans and some well-founded method to improve them, they can better develop advanced behaviors. Let us note that our point of view does not necessarily conflict with traditional approaches to learning, but it is complementary to them. In fact, simple plans could anyway be learned by using for

example reinforcement learning, instead of being pre-programmed, and then, by exploiting our technique, they could be improved and adapted for facing unknown situations.

The second assumption is justified by the existence of a consolidated research trend which outlines the need of modeling robot's cognitive activity for obtaining high-level reasoning capabilities. By following such idea, we have recently proposed a novel control architecture based on the concept of active mental entity. There are significant relationships between such concept and the notion of anticipatory system.

Here, we have illustrated how, by exploiting this architecture, sophisticated learning capabilities can be developed. As far as we know, other similar proposals do not exist yet. However, a recent work, implementing a *motivationally autonomous agent* (7), shares some of our basic considerations and motivations. Though the authors use words such as *planning* and *cognition* in a more limited sense with respect to us, we completely agree with their opinion that "a motivationally autonomous agent must have some memory of the past consequences of similar activities, and it must be capable of planning - i.e., it must use some form of cognition. Furthermore, […] it must want something, it must have goals" (7).

Our work can be considered as a step towards the satisfaction of such requirements. In fact, we provide an explicit model of robot's mental entities and show how robot operation, including a sophisticated cognitive activity, can be obtained by defining proper interaction mechanisms among mental entities. Such cognitive activity is also a good basis for implementing learning capabilities, which can be defined in a quite natural way starting from the interaction among mental entities.

Among the main advantages of our learning approach there is the fact that learning activity about a specific plan is carried on each time the plan is executed, independently from the actual intention underlying it. For instance, referring to our example, learning about *Plan1* goes on every time such plan is used to reach a target, independently from the actual target to be reached and from the initial robot position. Similarly, the results of learning activity are general and can be usefully exploited independently from specific context situations. Such independence from contextual details is very important: in a sense the robot learns the concept of obstacle, rather than learning how to avoid specific obstacles put in specific positions when it has to go along a specific path.

This high level of generality is hardly achieved by other approaches. For instance in the HAMSTER experiment, described by (6), the learning method is not able to combine obstacle avoidance with food searching and storing, so that learning procedures have been carried out in a simulated environment without obstacles, whereas the obstacle avoidance routine has been pre-programmed by the user. In (7), the robot learns optimal trajectories between fixed start and target points, so that what has been learned is no more useful if the target point is moved. Similarly, in (9) the robot adapts its trajectories in order to periodically pass over a fixed battery recharging point, rather than pursuing the goal of recharging itself when necessary.

Among the most important directions of future work we mention: a further and deeper analysis of the relationship between anticipatory capabilities and learning, the study of efficient heuristics for the plan review phase, and the definition of a more sophisticated method for evaluating plans in the consolidation phase.

REFERENCES

1. Baluja S., Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller, *IEEE Trans. on Systems, Man, and Cybernetics*, 26 (3), 450-463 (1996).
2. Baroni P., Fogli D., Guida G., Mussi S., Adding Active Mental Entities to Autonomous Mobile Robot Control Architectures. *Proc. of the 2nd Int. Conf. on Advanced Robotics, Intelligent Automation and Active Systems*, Wien, 142-148, 1996.
3. Baroni P., Fogli D., Guida G., Mussi S., An Advanced Control Architecture for Autonomous Mobile Robots: Modeling Intentions and Persuasions. *Proc. of the 4th Int. Conf. on Control, Automation, Robotics and Vision*, 853-857, 1996.
4. Baroni P., Fogli D., Guida G., Mussi S, Modeling the Mental Activity of an Autonomous Agent: An Implementation Based on Intentions and Persuasions, *Proc. of Cybernetics and Systems 98 14th European Meeting on Cybernetics and Systems Research*, Wien, A, 737-742, 1998.
5. Barto A. G., Sutton R. S., Watkins C. J. C. H., Learning and sequential decision making, *Learning and Computational Neuroscience*, M. Gabriel and J. W. Moore, eds., MIT Press, 539-602 (1990).
6. Colombetti M., Dorigo M., Borghi G., Behavior Analysis and Training - A Methodology for Behavior Engineering, *IEEE Trans. on Systems, Man, and Cybernetics*, 26 (3), 365-380 (1996).

7. Donnart J.-Y., Meyer J.-A., Learning Reactive and Planning Rules in a Motivationally Autonomous Animat, *IEEE Trans. on Systems, Man, and Cybernetics*, 26 (3), 381-395 (1996).
8. Dorigo M., Introduction to the Special Issue on Learning Autonomous Robots, *IEEE Trans. on Systems, Man, and Cybernetics*, 26 (3), 361-364 (1996).
9. Floreano D., Mondada F., Evolution of Homing Navigation in a Real Mobile Robot, *IEEE Trans. on Systems, Man, and Cybernetics*, 26 (3), 396-407 (1996).
10. Georgeff M. P., Lansky A. L., Reactive Reasoning and Planning. *Proc .of the 6th Nat. Conf. on Artificial Intelligence (AAAI-87)*, Seattle, WA, 268-272, 1987.
11. Holland J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI (1975).
12. Kröse B. J. A., Learning from delayed rewards, *Robotics and Autonomous Systems*, 15, 233-235 (1995).
13. Michel O., Khepera Simulator Package version 2.0: Freeware mobile robot simulator, University of Nice, 1996.
14. Millán J. del R., Rapid, Safe, and Incremental Learning of Navigation Strategies, *IEEE Trans. on Systems, Man, and Cybernetics*, 26 (3), 408-420 (1996).
15. Mondada F., Franzi E., Ienne P., Mobile robot miniaturisation: A tool for investigation in control algorithms. *3rd Int. Symp. on Experimental Robotics*, Kyoto, Japan, 1993.
16. Pfeifer R., Cognition - Perspectives from autonomous agents. *Robotics and Autonomous Systems*, 15, 47-70 (1995).
17. Pollack M. E. The uses of plans, *Artificial Intelligence*, 57 (1), 43-68 (1992).
18. Rosen R., *Anticipatory Systems - Philosophical, Mathematical and Methodological Foundations*, Pergamon Press, 1985.
19. Steels L., When are robots intelligent autonomous agents?, *Robotics and Autonomous Systems*, 15, 3-9 (1995).