Learning to Solve Complex Tasks for Reactive Systems (Extended Abstract)

Mario Martin and Ulises Cortés

Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028, Barcelona (Catalunya), Spain

1 Introduction

One important issue in reactive systems research is the learning ability to solve complex tasks. There has been several mechanisms proposed to make reactive systems learn, for instance [4] [5] (for a review see [2]). These mechanisms consist in obtaining, usually by means of a trial and error process guided by a reinforcement signal, a mapping between the set of possible perceptions and the set of actions, that describes an adequate behavior.

Unfortunately, these methods are not well suited to learn how to solve complex tasks. A task whose solution is formed by a long chain of actions can be roughly considered as complex. It is hard to solve by trial and error these tasks because the number of possible chains of actions to consider grows exponentially with the length of the chain solution.

We propose that this limitation can be overcome by previously learning to solve more simple but general tasks, useful to solve the initial problem. This approach implies a *constructivist* or *developmental* learning process in which the system steps over different stages. In each stage, characterized by the set of actions the system knows, it learns new behaviors that can be used as new complex actions to solve more complex tasks in a higher stage.

This article develops this proposal with a new reinforcement learning algorithm. The new algorithm is necessary due to the inadequacy of other known algorithms in learning general behaviors in our framework.

2 Learning to Solve Complex Tasks

In this section, the features for a good learning method to solve complex tasks are discussed and considered in order to build our learning mechanism.

When trying to develop a mechanism for learning to solve *complex tasks* by reinforcement, we must understand the difficulties they present. The complexity of a task is mainly conditioned by the number of actions that composes the solution (actions executed until a positive or negative reinforcement is obtained) and by the number of perceptual states the system must deal with.

Without any initial knowledge, the reactive system has only the trial and error procedure for trying to solve any problem. It is known that it is very hard *learn by trial and error how to accomplish goals which imply a long sequence of actions.* Then, the complexity of the task comes from the adequacy of the actions the system can take to solve it: an adequate set of general actions lets solving the problem in a few steps. The system, built with a set of primitive actions can find some tasks too difficult to be learnt from the beginning just with it. In our approach, the system solves this impasse learning to solve more general actions which will let it improve its skills. In order to learn these new actions, the system will try to learn how to solve intermediate general tasks. The behaviors learnt to solve these tasks can be included as new available actions to the initial set. The intermediate tasks the system must learn in order to solve a complex task will be given opportunely to the system by a teacher.

On the other hand, it is known from the debate Ginsberg-Chapman [1] that, for solving complex tasks, it is necessary to *generalize* situations. In complex tasks, the number of different situations is too large to consider a response for each of them. Chapman [1] shows that a limited world perception by means of "deictic" sensors can be indicated for reactive systems solving complex tasks, producing a very profitable generalization process. This limited perception implies that the information of the environment the system has, is always partial, *incomplete and ambiguous.* Most learning mechanisms, for instance Q-learning [5], suppose the system having complete information about its environment. When these methods have to deal with incomplete information or ambiguities, the problem of "perceptual aliasing" [6] appears, disabling the learning process. Then, an important feature that our learning process must show is the ability for learning with limited world perception in order to solve complex tasks.

Finally, the reinforcement is given to solve a concrete problem, but an important feature the learning mechanism must show is the ability of learning *general behaviors* (not particular solutions) to solve problems. Problems with the same general goal ought to be solved with the same learnt method. This feature is not usually fulfilled in reinforcement learning mechanisms. They usually learn how to solve a problem in a concrete environment.

3 A new learning mechanism

In this section, we expose the basic learning mechanism developed according to the requirements obtained in the previous section. This mechanism must be able to learn general behaviors and to deal with incomplete information. In order to build a hierarchy of behaviors, it also must show a good performance in learning *reliable* behaviors from a crisp reinforcement signal.

The architecture developed by the learning process consists in an indexed table. The entries of the table are formed by the state of the sensors (situation). The output consists in statistical information to compute the "probability" to obtain, for each executable action, a positive reinforcement. This statistical information is composed of the number of successes and failures occurred depending on the execution or not execution of each action in that situation.

The probability to get a positive reinforcement when executing an action a, given the actual situation s, is estimated as follows [1]:

$$P(s,a) = \frac{S\_A(s,a)}{S\_A(s,a) + F\_A(s,a)} - \frac{S\_NA(s,a)}{S\_NA(s,a) + F\_NA(s,a)}$$

$S\_A(s,a)$ is the number of successes when action a has been executed in situation s, $S\_NA(s,a)$ is the number of successes obtained when the action has not been executed, $F\_A(s,a)$ is the number of failures when the action has been executed, and $F\_NA(s,a)$ is the number of failures when the action has not been executed.

With all this considerations, the algorithm for learning behaviors will consist in the execution of the best action (defined as the one with a higher probability of leading to a positive reinforcement) for every situation faced until there's an error or a success. When this process has finished, the learning consists in to actualize and weight the statistics involved in the choosing of actions (see details in [3]). The learning stops when the system does not learn anything in a predetermined number of new problems presented. In this case, the knowledge of how to solve the task is kept in a policy table which will be used to increase the system's set of actions and to achieve solving more complex problems.

The proposed basic algorithm presents several differences respect to other well known reinforcement learning mechanisms, specially with the TD family ([4], [5]), mainly in the learning of reliable and general behaviors under perceptual aliasing (see details in [3]).

According to section 2, the possibility of learning to solve a complex task is subordinated to the set of actions available to the system. In order to achieve the needed skills for facing complex tasks, a teacher must guide the learning process giving a sequence of general tasks to be learnt.

For every task the teacher must know the following information: the objective state (when the positive reinforcement signal must be given), when an error has occurred (i.e. the maximum number of actions to be taken to solve a problem and the dangerous states) and finally, the set of useful actions to solve a task from the whole set of possible actions, as well as the subset of sensors.

From this knowledge, a hierarchy of behaviors is built up until the system can solve the desired complex task. An example of this construction is exposed in the next section.

4 Experiences

A set of experiments have been performed in order to test the performance of the learning mechanism. The environment selected is the "blocks world", that fulfills the required conditions of complexity: incrementally complex tasks can be proposed, the perceptual system gives incomplete or ambiguous reports, the

[1] This measure has been compared with other ones and presents a more successful performance. Particularly, this measure uses information about the number of successes when the action has not been activated, given a estimation more similar to a correlation that a conditional probability.

initial set of actions is too simple to solve complex tasks, etc. This environment has also been used previously by Chapman [1] and Whitehead and Ballard [6] to show the possibilities and drawbacks of reactive systems in complex domains.

The system has been designed for this environment with a set of sensors that reports ambiguous information of the world, and a very simple set of actions. The sensory system is composed of: foveal vision, peripheral vision, visual memory, proprioceptive sensors and tactile sensors. The initial set of actions of the system is composed of very primitive actions over the visual focus and over a mechanical hand (see details in [3]).

The complex task the system must solve consists in putting an unlocalized block of a given color in the position where a mark lies. Initially, the system only has the mentioned set of actions and sensors. Then, a solution to the problem will be composed of a large chain of actions that, as we have seen in section 2, is the main problem that prevents learning from a trial and error procedure.

In order to solve the problem with our method, we will suppose that the teacher provides a set of general subtasks to learn which allows to solve the initial problem. In our case, the teacher proposes to learn the following general tasks: Searching for a block, Moving the hand to the visual focus, Removing the top of a stack and Grasping a block. The learning process for each of these tasks generates a behavior that can be considered as a new general action. These new actions increase the repertory of actions of the system and can be used for solving more complex problems.

The experimental results (see details in [3]) are positive for the proposed task and indicates that complex tasks as the “fruitcake” problem [1] can be solved in this way.

5 Conclusions

This research has lead us to show that it is possible, for reactive systems, to learn how to solve complex tasks. The task proposed in the “blocks world”, considering the initial set of actions the system knows, is not currently resolvable by any other direct learning method. The success of our proposal is due to the use of a learning mechanism robust to ambiguous information, that can improve the abilities of the system, learning new behaviors to solve general tasks.

References

1. D. Chapman. Penguins can make cake. *AI Magazine*, 10:45–50, winter 1989.
2. L.P. Kaelbling. *Learning in Embedded Systems.* MIT Press, 1993.
3. M. Martin. Learning to solve complex tasks by reinforcement: A new algorithm. Technical report, Universitat Politècnica de Catalunya, 1995.
4. R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.
5. C. Watkins. *Learning from delayed rewards.* PhD thesis, Cambridge Univ., 1989.
6. S.D. Whitehead and D.H. Ballard. Learning to percive and act by trial and error. *Machine Learning*, 7:45–83, 1991.