Implementing Adaptive Capabilities on Agents that Act in a Dynamic Environment

Giuliano Armano
DIEE
Department of Electrical and Electronic Engineering
University of Cagliari
armano@diee.unica.it

Giancarlo Cherchi
DIEE
Department of Electrical and Electronic Engineering
University of Cagliari
cherchi@diee.unica.it

Eloisa Vargiu
CRS4
Center for Advanced Studies, Research and Development in Sardinia
eloisa@crs4.it

Abstract

Acting in a dynamic environment is a complex task that requires several issues to be investigated. In this paper, a life-cycle for implementing adaptive capabilities on intelligent agents is proposed, which integrates planning and learning within a hierarchical framework. The integration between planning and learning is achieved by an agent architecture explicitly designed for supporting abstraction. Planning is performed by adopting a hierarchical interleaved planning and execution approach. Learning is performed by exploiting a chunking technique on successful plans. A suitable feedforward neural network selects relevant chunks used to identify new abstract operators. Due to the dependency between abstract operators and already-solved planning problems, each agent is able to develop its own abstract layer, thus achieving an individual adaptation to the given environment.

1. Introduction

Using intelligent agents in applications built to work in real-world domains has introduced several issues to be investigated; in particular: (i) how to devise a suitable *planning* strategy able to deal with the underlying complexity, (ii) how to implement the capability to react promptly to environmental changes, and (iii) how to enable agents to adapt themselves to the given environment.

An effective approach adopted for dealing with the complexity of the planning task is to build a set of abstractions so that the search can be controlled by mapping the original search space into new abstract spaces, in which irrelevant details have been disregarded at different levels of granularity. A hierarchical planner first solves a problem in the most abstract space and then uses the abstract solution to guide the search in the underlying space. The search proceeds recursively until a solution is found while searching the ground space.

To give an agent the capability to react to any change in the environment, interleaving planning and execution monitoring can be adopted (e.g. [1] and [4]). In this way, all events that make the current activity needless or impossible to be completed can be quickly identified. Furthermore, whenever an agent does not have complete information about its environment, an early execution of the actions could be used to obtain more information[5].

Self adaptation is a very powerful tool for agents acting in complex environments. In this paper, an adaptive mechanism that allows an agent to discover abstract operators from successfully solved plans is described. Since planning is performed exploiting two levels of abstraction (i.e. *ground* and *abstract*), an abstract solution can be seen as a sequence of abstract operators, each embodying a subproblem to be solved by the underlying, ground-level, planner. Any new abstract operator becomes a candidate for being embedded into the abstract planner for other problems to be solved.

2. A hierarchical adaptive approach for acting in a dynamic environment

A hierarchical interleaving planning and execution approach (HIPE) has been adopted for designing and implementing agents, which encompasses three main interacting aspects: (i) their underlying architecture and domain knowledge, (ii) their proactive and reactive capabilities, and (iii) their adaptive behavior.

The proposed architecture is two-pass vertically layered and can be provided with N layers. Each layer exploits a local knowledge base (KB), and is numbered according to its level of abstraction. All layers are conceptually identical, each being able to embody in the same way reactive, deliberative and proactive functionality.

We implemented agents provided with two layers (i.e., *situated* and *strategic*), each one capable of hosting a corresponding level of abstraction (i.e., *ground* and *abstract*, respectively), and provided with a planner, based on the UCPOP algorithm [6].

An agent repeatedly performs planning, executes actions, and monitors any change in the environment, both at the situated and strategic layer. The ground-level planner performs planning on any goal imposed by the abstract-level planner, so that "executing" an action at the abstract level actually means creating a subgoal to be solved by the ground-level planner. Plan execution starts as soon as the first abstract operator has been refined. On completion, the next abstract operator (if any) is refined, according to a depth-first approach (see [2] for further details).

Two main problems arise, concerning the soundness and completeness properties of an abstraction. Here, we are interested in dealing with a "quasi-sound" abstraction, i.e., with an abstraction in which the near-DRP holds (see [3]). On the other hand, no a-priori assumptions are made about completeness, except for the fact that it could be asymptotically obtained by repeatedly adding abstract operators to the planner located at the strategic layer.

To be effective, the proposed HIPE approach requires an adaptive mechanism to be enforced. Here, we propose an automated mechanism for generating abstract operators (useful for any subsequent search at the abstract level), given an initial (hand-coded) hierarchical description of the domain, together with a set of already solved problems and their corresponding solutions. Successful plans are analyzed by the learning algorythm, particularly those for which the abstract-level planner failed, in search of relevant sequences that could play the role of "supporting macro-operators" while devising new operators at the abstract level.

2.1. The adaptive behavior life cycle

To identify abstract-operators, a "chunking" technique is exploited, which extracts sequences from successful plans. Relevant sequences are then identified by a feedforward neural network (FFNN), fed by a vector of suitable metrics evaluated for each given sequence. The adaptive life-cycle consists of repeatedly performing the following actions:

1. The HIPE planner is activated on the given goal (let us assume that a suitable solution is found and executed).
2. Plan subsequences are randomly extracted from the plan (chunking).
3. Relevant sequences are filtered out by a suitable FFNN. For every sequence, a corresponding vector of metrics is evaluated, to be used as input to the FFNN.
4. Each sequence deened relevant passes through a schematization process, which creates a corresponding macro-operator schema (MO). All MOs are temporarily stored in the MOS REPOSITORY.
5. Each MO passes through an abstraction process, which creates a corresponding abstract-operator schema (AO) from any given MO. For every MO, the AO REPOSITORY is updated
6. When an AO is stored in the AO REPOSITORY, a suitable procedure must be invoked that decides whether or not the AO has to be added to the set of AOs (actually used by the abstract planner).

3. Experimental results

The system has been implemented as a prototype for a computer game, in which the user plays in a virtual city populated by different kinds of entities, all implemented by intelligent agents. Players have an agent representative within the game (i.e., an avatar), which can be given a goal to be attained through a simple graphical interface.

Steps 1 to 5 of the adaptive life-cycle previously described have already been implemented. As for step 6, we are currently searching for an optimal balance between the augmented capability of solving problems at the abstract level and the growth of the search complexity due to the higher branching factor.

The FFNN adopted for identifying relevant sequences is composed of an input layer of 7 neurons, together with a single hidden layer of 3 neurons, and has been trained with sequences extracted from a successful plan generated by the HIPE planner. A threshold μ_A has been used to separate relevant sequences from non-relevant ones and about 97% of the training sequences have been recognized as belonging to their proper class. The test set we used has been taken from typical virtual world situations. The FFNN identified several sequences (all considered relevant) useful for abstraction and not yet obtained as a refinement of an existing abstract operator. Starting from these sequences, new abstract-operator candidates have been successfully generated by the system.

4. Conclusions and future work

In this paper, a first step towards agents that can develop an adaptive behavior has been performed. The adaptive life-cycle, integrated within a hierarchical interleaving planning and execution approach, is driven by the given environment and by the history of problems that have been submitted to each agent.

As for future work, we are testing the potential of domain ontologies for integrating planning and learning within a unifying framework. Furthermore, we are investigating how to automatically assess when a candidate abstract operator can be embodied in the set of active abstract operators.

References

[1] J. A. Ambros-Ingerson, and S. Steel, "Integrating Planning, Execution and Monitoring", *Proc. of the 7th National Conference on Artificial Intelligence*. 1988, pp. 83-88.

[2] G. Armano, and E. Vargiu, "An Adaptive Approach for Planning in Dynamic Environments", *2001 International Conference on Artificial Intelligence, IC-AI 2001, Special Session on Learning and Adapting in AI Planning,* Las Vegas, Nevada, June 25-28, 2001, pp. 987-993.

[3] F. Bacchus, and Q. Yang, "Downward Refinement and the Efficiency of Hierarchical Problem Solving", *Artificial Intelligence*, Vol. 71(1), 1994, pp. 41-100.

[4] K. Z. Haigh and M. Veloso, "Interleaving Planning and Robot Execution for Asynchronous User Requests", *Autonomous Robots*, Vol. 5(1), March 1998, pp. 79-95.

[5] I. Nourbakhsh, *Interleaving Planning and Execution for Autonomous Robots*, Kluwer Academic Publishers, 1997.

[6] D. Weld, "An Introduction to Least Commitment Planning", *AI Magazine*, 1994, pp. 27-61.