Dynamic Abstraction for Hierarchical Problem Solving and Execution in Stochastic Dynamic Environments

Per Nyblom
Department of Computer and Information Science, Linköping university, Sweden,
email: perny@ida.liu.se

1. Introduction

Most of today's autonomous problem solving agents perform their task with the help of problem domain specifications that keep their abstractions fixed. Those abstractions are often selected by human users.

We think that the approach with fixed-abstraction domain specifications is very inflexible because it does not allow the agent to focus its limited computational resources on what may be most relevant at the moment. We would like to build agents that dynamically find suitable abstractions depending on relevance for their current task and situation. This idea of *dynamic abstraction* has recently been considered an important research problem within the area of *hierarchical reinforcement learning* [1].

2. Algorithm

We have developed an algorithm that is designed to use techniques for dynamic abstraction targeted at adaptive problem generation. It operates in a hierachical manner and interleaves problem solving and execution. The algorithm is a template and needs to be augmented with domain-specific dynamic abstraction methods, solution techniques and subproblem generation. The algorithm uses a dynamic hierarchy of solutions which is called a Hierarchical Solution Node (HSN) structure where each node represents an abstraction level with a corresponding problem model. A HSN structure is somewhat similar to the task graph used in the MAXQ value function decomposition [2]. Replanning and modifications to the problem model abstraction are performed continuously depending on whether the current abstractions are considered invalid or not, which makes the HSN structure change dynamically. State changes may trigger creation of new subtasks. This results in a new problem model abstraction and replanning for that particular task.

3. Problem Domain and Implementation

We have implemented the algorithm for a domain inspired by our unmanned aerial vehicle (UAV) research [3], where we show how dynamic abstraction can be done in practice.

The domain consists of a freely moving agent in a continuous 2D environment without obstacles. The agent's task is to maximize its total reward which is increased by classifying moving targets and to finish at so called *finish areas*. The reward is decreased when the agent comes too close to any of the moving dangers in the environment. The movement of the targets and dangers is stochastic and they can either be contrained to move on a road network or operate freely.

A simple fixed abstraction scheme will eventually fail in this domain due to the curse of dimensionality when the number of objects increases. We have therefore developed a method to find suitable abstractions dynamically.

The problem model abstractions are in this case the possible discretizations of the different features in the domain such as the position features of the dangers and targets. The discretizations are limited by a maximum state space size. A utility measure for discretizations, based on the features' expected relevances in the current situation, are used to state the abstraction selection as an optimization problem. For example, the relevance for a danger's position feature depends on its distance from the agent and its ability to inflict negative reward. Each feature's utility increases with the number of discrete values it can take in the final discretization. The total utility, which is the measure that is maximized, is the sum of all the features' utility functions.

The optimization problem is solved by hillclimbing in our current implementation and results in a specification of how many states that each feature should get in the final discretization. The state space is then divided by k-means clustering together with the agent's internal simulation model of the environment.

The same simulation model is used to solve the problem that is defined by the discretization with the DynaQ [4] model-based reinforcement learning algorithm.

When a problem is solved on the selected abstraction level, subtasking is performed by creating a new subproblem that corresponds to the first step in the solution.

Replanning is performed either when the abstraction is considered too old or when the relevances of the features in the current state differs too much from the ones used in the abstraction.

Experiments with our implementation indicate that the abstractions must be replaced frequently for good results in this particular domain. It might therefore be more efficient to use a forward search method instead of reinforcement learning for solving the problems.

References

[1] A. G. Barto and S. Mahadevan, 'Recent advances in hierarchical reinforcement learning.', *Discrete Event Dynamic Systems*, **13**(4), 341–379, (2003).

[2] T. Dietterich, 'Hierarchical reinforcement learning with the MAXQ value function decomposition', in *Proceedings of the 15th International Conference on Machine Learning*, (1998).

[3] P. Doherty, 'Advanced research with autonomous unmanned aerial vehicles', *Proceedings on the 9th International Conference on Principles of Knowledge Representation and Reasoning*, (2004).

[4] R. S. Sutton, 'Integrated architectures for learning, planning, and reacting based on approximating dynamic programming', in *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224, (1990).