Hierarchical Skill Learning for High-level Planning

James MacGlashan *
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore, Maryland
jmac1@cs.umbc.edu

Introduction

Many of the existing techniques for controlling goal-directed agent behavior use one of two primary approaches: heuristic-search planning (HSP) or reinforcement learning (RL). Each has its advantages and disadvantages. For instance, heuristic-search planning does not traditionally learn from previous experience, and can only be applied in domains for which a complete domain model exists. On the other hand, reinforcement learning often performs poorly in new situations until it has gained enough experience to learn an effective policy, and it is difficult to scale RL up to large, complex domains.

Both RL and HSP tend to work poorly in domains that require long action sequences. Heuristic-search state-space planning is intractable in such domains, because of the very large search spaces, and reinforcement learning may require exponentially many execution traces to converge. For agents with only low-level primitive actions, such as moving limbs, this makes it intractable to solve problems in complex domains.

In order to address both the unique and the shared problems of HSP and RL, I propose a new research direction called skill bootstrapping (SB). The goal of SB is to provide an integrated learning and planning architecture that can improve its performance over time in complex domains. An SB agent starts with a basic set of primitive actions (and their preconditions and effects) as its model of the world. Over the course of solving numerous problems by applying HSP to the primitive actions, SB identifies recurring subgoals, for which it uses RL to create skills that can be applied within the HSP process to solve these subgoals more efficiently. Subgoals can be set by a human supervisor, by a request from the environment, or through a process of exploration. The skills behave as partial policies that can be used reactively, without lengthy deliberative reasoning.

Once a new skill is learned, it becomes available for use by the planner along with the other primitive actions, allowing for more compact plans. Additionally, just as future plans can use learned skills, future skills may be built upon lower-level skills. Over the course of the agent’s experience, this process will eventually result in a hierarchy of skills that support high-level reasoning.

I have developed a framework for SB that consists of three main components: planning, skill identification, and skill learning. The planning component creates plans to solve each problem that an agent encounters. Plans created by the planning component may consist of primitive actions, skills, or a mixture of both.

The skill identification component stores a library of successful plan traces. Using this library, plan traces are compared with each other to find sets of plans that have the same type of goal, but are not necessarily identical. When such a set of plans is discovered, a skill to solve the common goal is created.

Once the skill identifier creates a new skill, it is passed to the skill learning component so that the skill’s policy may be learned. Initially, learning is seeded by the set of plan traces from which the skill was identified. Skill learning is achieved using Q-learning function approximation, so that the resulting policies can be applied to goals and states that have not been previously explored.

Each of these three components represents a set of research challenges to be solved. The planning component must be able to detect when a skill has failed, since there may not necessarily be a failed terminal state. Skill failure may be detected when the confidence associated with a skill’s function approximation value is low, or when the skill takes an abnormally long time to complete.

For the skill-identification component, one challenge is how to organize the plan library so that it may be quickly checked for plans with the same type of goal. Identifying goals of the same type may also be challenging. This problem might be solved by organizing goals by object type and properties, similar to how CHEF identifies similar planning problems (Hammond 1989). Additionally, the skill identification component must also determine how to parameterize a problem. This may be accomplished by determining which objects are affected in the source plan traces.

A challenge for the skill learning component is that it must apply function approximation to problems involving a variable number of objects and information. For instance, in a blocks world, a skill for grabbing a block should work in problems with either three or four blocks. However, function approximation in RL typically uses a feature vector with
a fixed dimensionality, which is inflexible to such problem variability. To resolve this situation, a function approximation technique such as locally weighted averaging (LWA) may be used, provided that a state similarity metric can be computed and provided as an input. I am currently investigating a metric that uses techniques from the information retrieval literature to compute state similarity.

**Related Work**

The concept of policy control that builds on lower-level action primitives to achieve a goal is not new to agent control. In planning fields, this notion is usually referred to as *macro-operators* or *macro-actions*. Macro-actions are generally constructed as a fixed sequence of primitive actions. These macro-actions may be constructed by analyzing the domain (Botea et al. 2005), using genetic algorithms (Newton, Levine, and Fox 2005), or with learning techniques (Coles and Smith 2007; Coles, Fox, and Smith 2007).

Two key commonalities of these approaches are that macro-actions consist of fixed sequences of actions, and that the list must be pruned to avoid large collections of macro-actions. The SB approach proposed here differs in that skills are policy control mechanisms that vary the action sequence depending on the particular state of the world.

Marthi et al. (2008) propose a hierarchical planner, AHA*, using high-level actions (HLAs) to specify partial plans that are refined in the planning process. Planning can take place at a high level by constructing abstract lookahead trees. A key difference from the SB approach is that SB creates plans using skills—reactive learned policies for solving subgoals—whereas AHA* must plan at every step.

Using hierarchies of actions in reinforcement learning has also been an area of active research. The MaxQ algorithm (Dietterich 2000) used a designed hierarchy of sub-tasks to efficiently solve more complex problems. Recent work has focused on automatically identifying the action hierarchy (Jonsson and Barto 2005; Mehta et al. 2008). With these algorithms, the action hierarchies are fixed structures that have a defined root structure and are specific to a single problem. With the SB architecture, skills learned in the SB architecture are not explicitly structured. That is to say, skills that are referenced by a parent skill do not have to be invoked by the parent. Rather, any skill can be independently invoked if it is pertinent to the problem at hand.

Other work on forming action abstraction comes from Simsek and Barto (2007), who examine the graph structure of problems to identify states that are likely to be important. The SB architecture differs in that instead of explicit states being used to identify places for skills, types of goals are identified that may result in different states that share similar properties and may therefore be parameterized.

**Research Schedule**

My research schedule is as follows:

2. Novel state similarity metric that can be used with parameterized RL in problems with a variable number of objects. Compare results to problems with fixed dimensionality. (5/10).
3. Planning and replanning with single-layer skills. Compare computational efficiency and plan quality to planning without skills (5/10).
4. Seeding skill learning with plan traces. Compare the learning speed to learning without seeding (8/10).
5. Learning and planning with hierarchical skills, using hierarchies constructed by a human. Compare efficiency and plan quality to planning without skill hierarchies (12/10).

**References**


