

---

# Hierarchical Reinforcement Learning Based on Automatic Discovery of Subgoals and Specialization of Subpolicies

---

Bram Bakker<sup>1,2</sup>

Jürgen Schmidhuber<sup>1</sup>

BRAM@IDSIA.CH , BRAM@SCIENCE.UVA.NL

JUERGEN@IDSIA.CH

<sup>1</sup> IDSIA, Galleria 2, 6928 Manno-Lugano, Switzerland

<sup>2</sup> IAS, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

## Abstract

We introduce a new method for hierarchical reinforcement learning. High-level policies automatically discover subgoals; low-level policies learn to specialize for different subgoals. Subgoals are represented as desired abstract observations which cluster raw input data. An experiment shows that this method outperforms several flat methods.

## 1. Introduction

The promise of scaling up reinforcement learning (RL) through *hierarchical* RL (HRL) is widely acknowledged. The idea is that low-level policies, which emit the actual “primitive” actions at a fast timescale, should solve only parts of the overall task. Higher-level policies should work on a slower timescale, solving the overall task by sequentially invoking lower-level policies, considering only a few abstract high-level observations and actions (macro-actions, options). Thus each level’s search space is reduced, temporal credit assignment is facilitated, and low-level policies are easily reusable within the task and in different tasks.

In most previous HRL studies the hierarchical structure was prewired by a designer. To minimize the latter’s responsibility, however, we would like to learn the hierarchy itself as well. This work presents a step in that direction. We propose the HASSLE algorithm, in which high-level policies automatically discover subgoals, and low-level policies learn to specialize for different subgoals.

## 2. HASSLE

In the HASSLE algorithm (Hierarchical Assignment of Subgoals to Subpolicies LEarning algorithm), each action of a higher-level policy  $\pi^H$  at each high-level time step  $t^H$  is the selection of a subgoal. The subgoal  $o_g^H$  is taken from the set  $o^H$  of abstract high-level ob-

servations which differs from the set  $o^L$  of “primitive” low-level observations. High-level observations basically classify low-level observations into a limited set of clusters, e.g. using an unsupervised learning clustering algorithm. The input  $o_s^H$  is also taken from the set  $o^H$ .  $\pi^H$  can learn no matter whether the desired subgoal  $o_g^H$  or a different  $o^H \neq o_g^H$  was reached. In the latter case it simply learns as if the reached  $o^H$ ,  $o_r^H$ , was the desired subgoal. A standard value function-based RL algorithm, in our case Advantage learning, can be used. In the table-based case, the update of the Advantage value  $A^H(o_s^H, o_r^H)$  corresponds to

$$\Delta A^H(o_{s,t^H}^H, o_{r,t^H}^H) = \alpha^H [V^H(o_{s,t^H}^H) + r_{t^H} + \gamma^H V^H(o_{s,t^H+1}^H) - V^H(o_{s,t^H}^H)] - A^H(o_{s,t^H}^H, o_{r,t^H}^H)$$

where  $V^H(o_{s,t^H}^H) = \max_{o_g^H} A^H(o_{s,t^H}^H, o_g^H)$ , and  $\alpha^H$ ,  $\gamma^H$ , and  $\kappa^H$  are constants.

It is the job of the low-level policies to reach the subgoal. There is a limited set of low-level policies  $\pi_i^L$ . Every  $\pi_i^L$  contains a table of so-called C-values of  $(o_s^H, o_g^H)$  pairs, each of which represents the “Capability” of  $\pi_i^L$  to reach subgoal  $o_g^H$  from  $o_s^H$ . Following a high-level observation change, one low-level policy is selected based on the C-values of all low-level policies for the current  $(o_s^H, o_g^H)$  pair (using a Boltzmann rule). This one then attempts to reach the current subgoal  $o_g^H$ . If it does indeed reach  $o_g^H$ , its C-value for this  $(o_s^H, o_g^H)$  pair is increased, making future selection of this low-level policy in this context more likely; otherwise the C-value is decreased. The update rule is

$$\Delta C_i(o_{s,t^H}^H, o_{r,t^H}^H) = \alpha_r^C [\gamma_C^{t_r^L - t_s^L} - C_i(o_{s,t^H}^H, o_{r,t^H}^H)]$$

where  $\alpha_r^C$  and  $\gamma_C$  are parameters,  $t_s^L$  is the low-level time step at which the current low-level policy was started, and  $t_r^L$  the one at which  $o_r^H$  was reached. In addition, if the subgoal was not reached,

$$\Delta C_i(o_{s,t^H}^H, o_{g,t^H}^H) = \alpha_n^C [0 - C_i(o_{s,t^H}^H, o_{g,t^H}^H)].$$

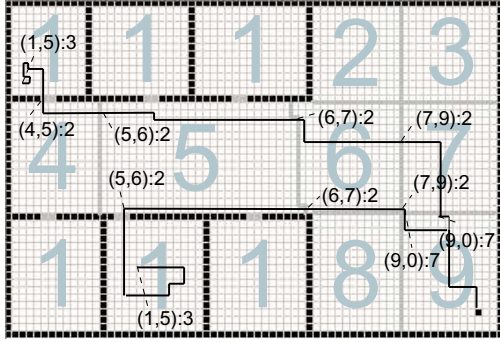


Figure 1. The office task. The goal is the black dot in the lower right corner. There are 6 rooms, connected by doors to a central corridor. Large grey numbers indicate 9 possible high-level observations (HLOs).

So a low-level policy may learn that from some start situation it can reach subgoal  $A$  but not subgoal  $B$ . Another low-level policy may learn the opposite. This realizes *specialization*. But a single low-level policy may also learn that its capability encompasses multiple  $(o_s^H, o_g^H)$  pairs. This realizes *generalization*. The idea is that the low-level policies will specialize when they have to, but generalize when they can.

To facilitate generalization and specialization, in the experiments we use (linear) function approximators for low-level policies. In this way, each low-level policy can learn to focus on those parts of the low-level observation space which are relevant to its specialization. The low-level policy again learns using Advantage learning. The weight updates at each low-level time step  $t^L$  for the currently active low-level policy  $\pi_i^L$  are

$$\Delta w_{i,m} = \alpha^L [V_i^L(o_{t^L}^L) + \frac{r_{t^L}^L + \gamma^L V_i^L(o_{t^L+1}^L) - V_i^L(o_{t^L}^L)}{\kappa^L} - A_i^L(o_{t^L}^L, a_{t^L}^L)] \frac{\partial A_i^L(o_{t^L}^L, a_{t^L}^L)}{\partial w_{i,m}}$$

where  $V_i^L(o_{t^L}^L) = \max_{a^L} A_i^L(o_{t^L}^L, a^L)$  and  $\alpha^L$ ,  $\gamma^L$ , and  $\kappa^L$  are constants. In the experiments we set  $r^L = 1$  if the subgoal is reached and  $r^L = 0$  otherwise.

### 3. An illustrative experiment

The experiment is a navigation task in an “office” grid-world. The agent should learn to move from random start positions to a fixed goal position (see Figure 1). It has an orientation and three primitive low-level actions: make a step in the current direction, turn left  $90^\circ$ , turn right  $90^\circ$ . Each episode ends once the agent reaches the goal, where it receives the only reward  $r = 4$ . Low-level observation vectors contain information from 4 “sonar” sensors which measure the distance to the nearest wall/door, 4 additional directed

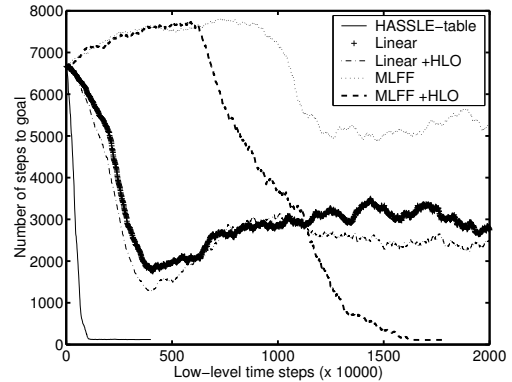


Figure 2. Average numbers of low-level actions needed to reach the goal, as functions of learning time.

sensors which detect the presence of doors, and four more which detect the goal, provided it is at most 8 grid cells away. The high-level observations are produced online by applying a simple unsupervised learning vector quantization method to the 4-dimensional output of the distance sensors. Figure 1 shows how it distributes high-level observations over the state space (large grey numbers). There were 8 low-level policies (a fairly arbitrary value; many other values work too).

We compared the HASSLE system to 4 flat RL systems. The first was a single linear function approximator with the low-level observations as inputs and trained with Advantage-RL. The second was a nonlinear function approximator, a multilayer feedforward neural network (MLFF). The final two systems were the same as the first two, but now the input vectors contained also  $o_s^H$  as computed by the vector quantizer. The idea here is to investigate the utility of one HASSLE aspect, namely spatial and temporal abstraction of observations.

Figure 2 plots the average number of primitive actions needed to reach the goal as a function of the total number of low-level time steps (averages of 10 runs). HASSLE converges to near-optimal policies. The other systems either fail completely, or take much longer to reach good solutions (MLFF with the high-level observation, HLO, as an additional input).

Figure 1 shows trajectories of a HASSLE agent after learning. The invocation of new subgoals and subpolicies is indicated by corresponding  $(o_s^H, o_g^H)$ : *subpolicy index* labels. For example, in the upper left room,  $o_s^H = 1$ , subgoal  $o_g^H = 5$  is selected, and specialized subpolicy 3 is started, which has learned to look for doors and go through them. Following the trajectories, one can see examples of specialization and generalization of the subpolicies, and how  $\pi^H$  selects proper subgoals following each high-level observation change.