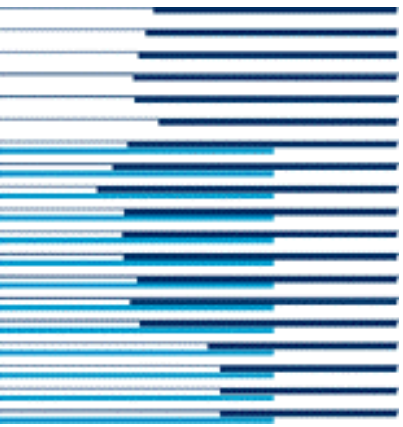
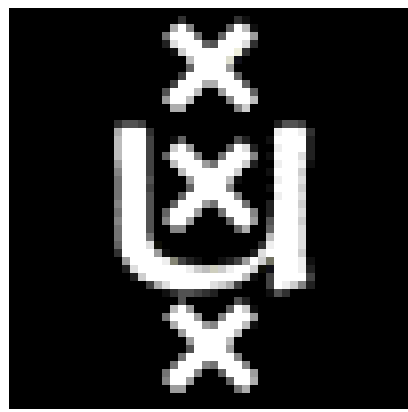
Hierarchical Reinforcement Learning Based on Subgoal Discovery and Subpolicy Specialization



Bram Bakker & Juergen Schmidhuber
IDSIA, Lugano, Switzerland
and
University of Amsterdam, The Netherlands



EWRL 6, Nancy, Sept. 2003

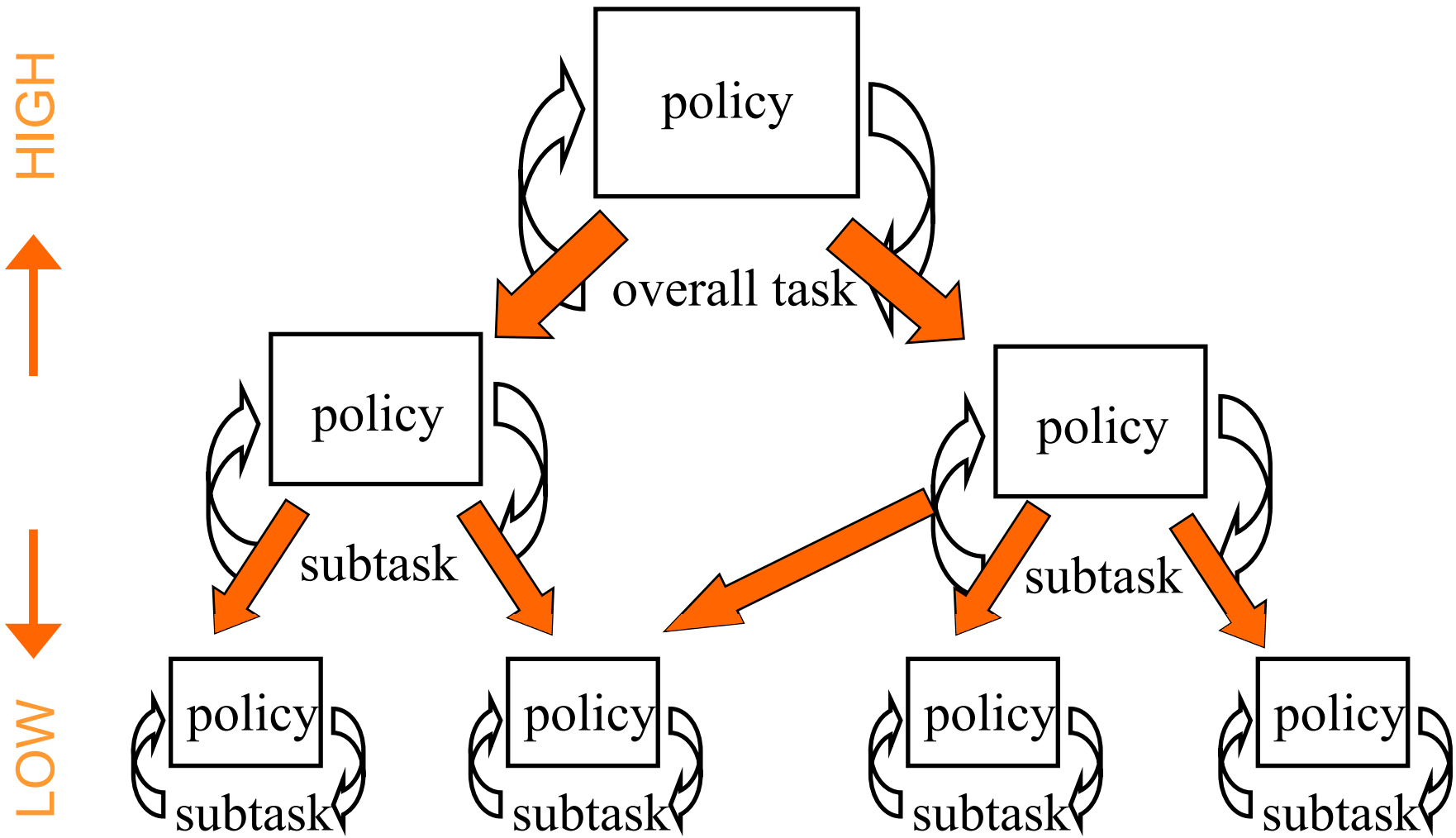
Outline of the talk

- Hierarchical Reinforcement Learning (HRL): advantages and disadvantages
- Our new HRL algorithm that learns the hierarchical structure
- An illustrative experiment: office navigation
- Results and comparisons to flat RL systems
- Conclusions and future work

Problems for standard, “flat” RL

- Large state-action spaces
- Long temporal credit assignment paths

HRL basic idea



HRL advantages

- Reduces the search space for all levels in the hierarchy
  - Low-level policy (subpolicy) only solves subtask, i.e. part of the overall task
  - High-level policy only considers abstract, high-level observations and actions, at slower temporal resolution
- Facilitates temporal credit assignment for all levels
- Subpolicies can be used over and over again in the same task and other tasks
- Facilitates building in knowledge

HRL disadvantages

- Final hierarchical policy may not be the optimal policy
- Convergence (even) harder to prove
- Decomposition into “good subtasks” is not trivial

Challenge: learning the hierarchy

- Most hierarchical RL work assumes a prewired hierarchical structure (MAXQ, Options, Feudal Q)
- Can we learn the hierarchy itself, i.e. the decomposition into useful subtasks and useful subpolicies? (HQ, HEXQ, Nested Q, SSS)

"A key open question is how to form task hierarchies automatically" (Barto & Mahadevan, 2003)

HASSLE algorithm: basic idea

- HASSLE: **H**ierarchical **A**ssignment of **S**ubgoals to **S**ubpolicies **LE**arning

- High-level policies learn to identify *subgoals* associated with high-level observations
- Simultaneously, low-level policies learn to *specialize* for different subgoals

High-level policy

- Every high-level timestep (t^H) corresponds to a high-level observation (o^H) change
- Every t^H, using the current $o^H{}_S$ as input, the high-level policy selects a *subgoal* $o^H{}_G$, i.e. the o^H that it wants to see next
- Learning using standard RL algorithm

Low-level policies (1)

- Every low-level timestep (t^L) corresponds to a low-level observation (o^L) change
- Limited set of low-level policies
- Every low-level policy contains a table of *C-values* $C(o^H_S, o^H_G)$, representing the "Capability" to reach subgoal o^H_G from o^H_S
- Based on the C-values, one low-level policy is selected and tries to reach the subgoal

Low-level policies (2)

- If the active low-level policy reaches the subgoal
  - increase $C(o^H_S, o^H_G)$
  - reward the low-level policy itself, which learns using standard RL algorithm
- Otherwise:
  - decrease $C(o^H_S, o^H_G)$
  - punish the low-level policy

Low-level policies (3)

- Low-level policies will *specialize* for subgoals when they have to, but *generalize* when they can (more so than e.g. Wiering's HQ and Sun's SSS)
- Low-level policies use function approximators, such that each can focus on specialized parts of low-level observation space

Learning rules

- High-level policy:

$$\Delta A^H(o^H_{s,t^H}, o^H_{r,t^H}) =$$

$$\alpha^H[V^H(o^H_{s,t^H}) + \frac{r_{t^H} + \gamma_H V^H(o^H_{s,t^H+1}) - V^H(o^H_{s,t^H})}{\kappa^H} - A^H(o^H_{s,t^H}, o^H_{r,t^H})]$$

- Low-level policies:

$$\Delta w_{i,m} =$$

$$\alpha^L[V^L_i(o^L_{t^L}) + \frac{r^L_{t^L} + \gamma_L V^L_i(o^L_{t^L+1}) - V^L_i(o^L_{t^L})}{\kappa^L} - A^L_i(o^L_{t^L}, a^L_{t^L})]\frac{\partial A^L_i(o^L_{t^L}, a^L_{t^L})}{\partial w_{i,m}}$$

- C-values
  - when subgoal is reached:

$$\Delta C_i(o^H_{s,t^H}, o^H_{r,t^H}) = \alpha^C_r[\gamma_C^{t^L_r - t^L_s} - C_i(o^H_{s,t^H}, o^H_{r,t^H})]$$

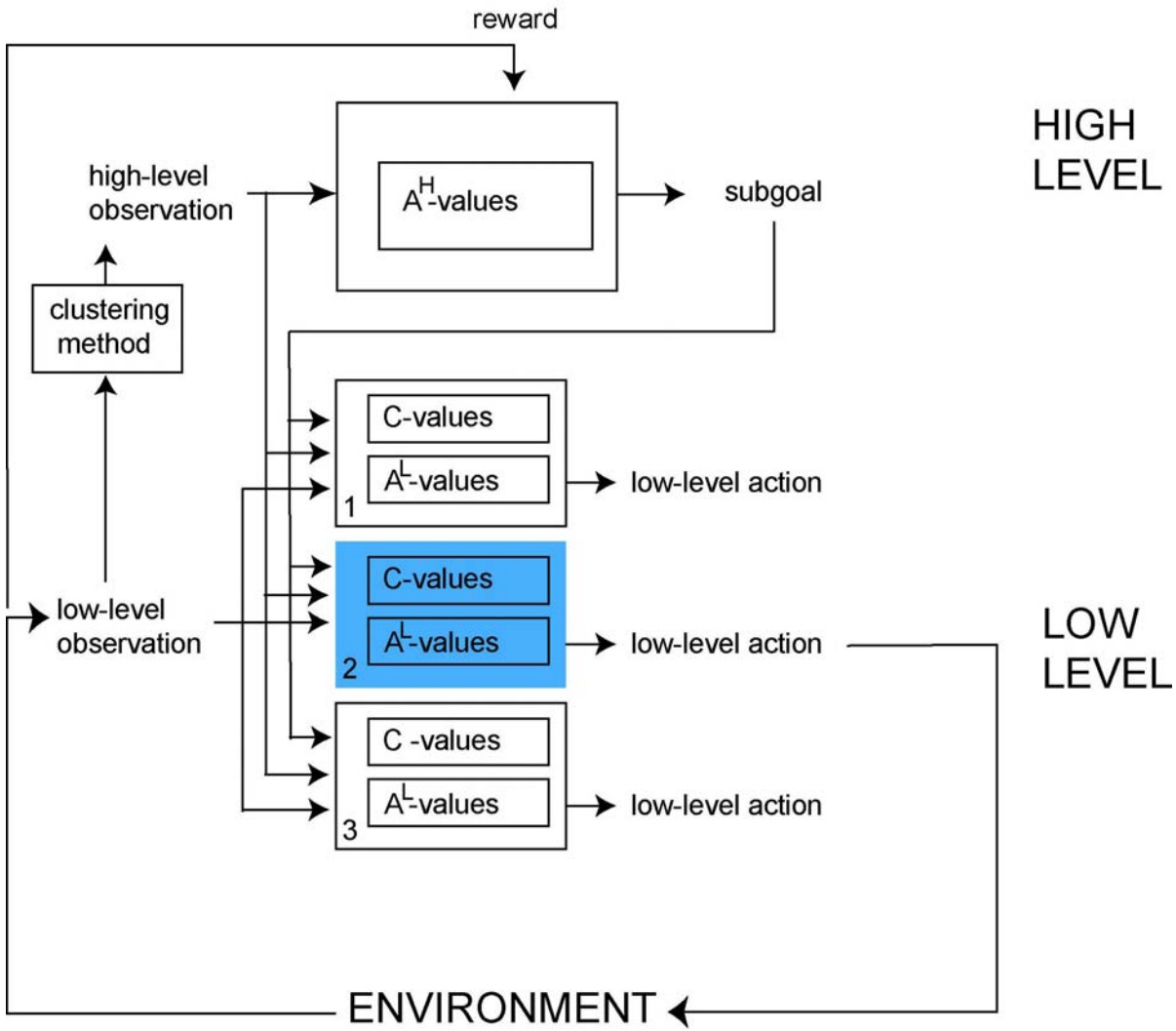
  - when subgoal is not reached:

$$\Delta C_i(o^H_{s,t^H}, o^H_{g,t^H}) = \alpha^C_n[0 - C_i(o^H_{s,t^H}, o^H_{g,t^H})]$$

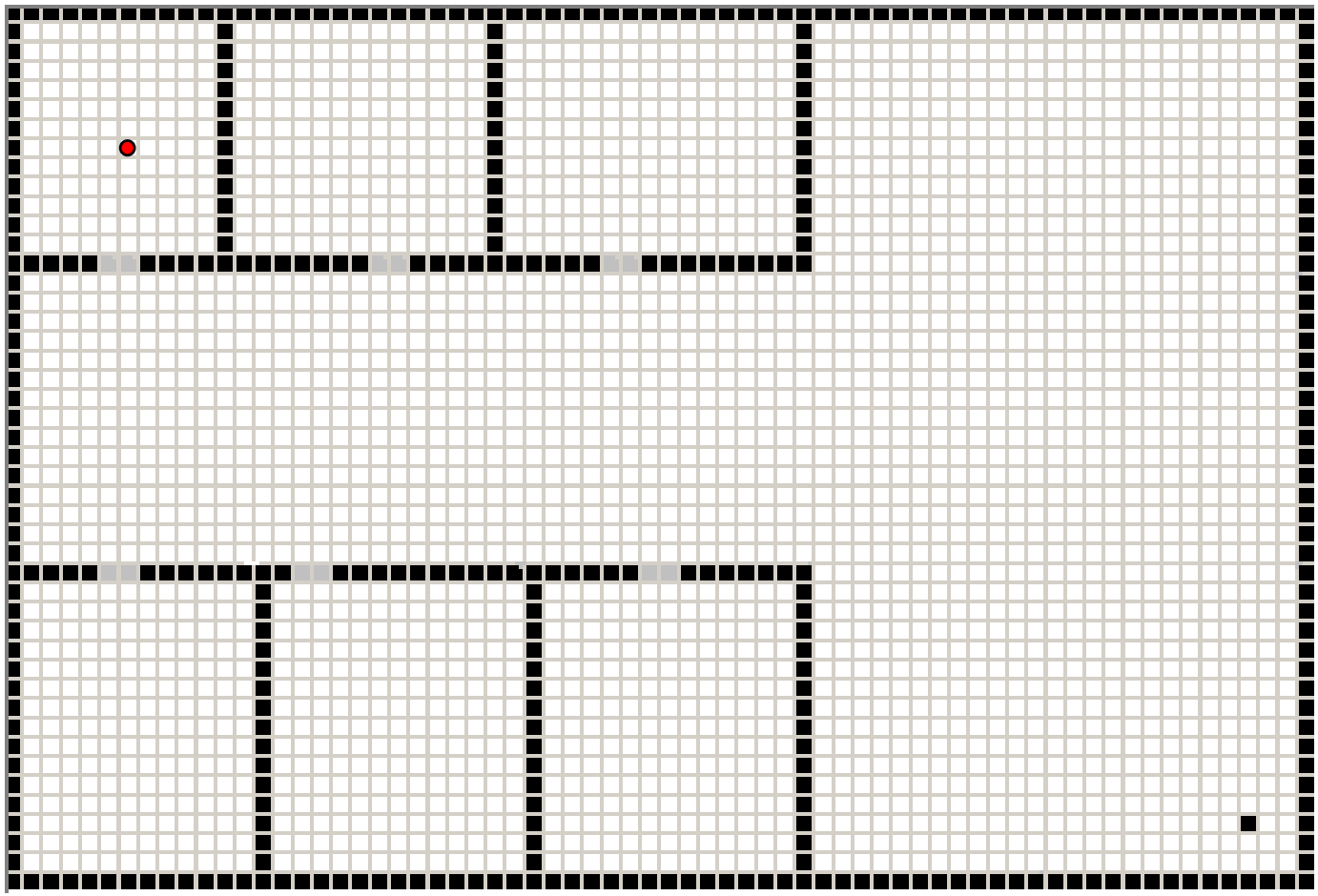
Producing high-level observations

- HASSLE is not constrained to particular method
- High-level observations should cluster raw low-level observations such that neighboring states are usually clustered together
- In the experiments we used a simple unsupervised learning vector quantization algorithm

HASSLE schematic



Experiment 1: office navigation



Office navigation

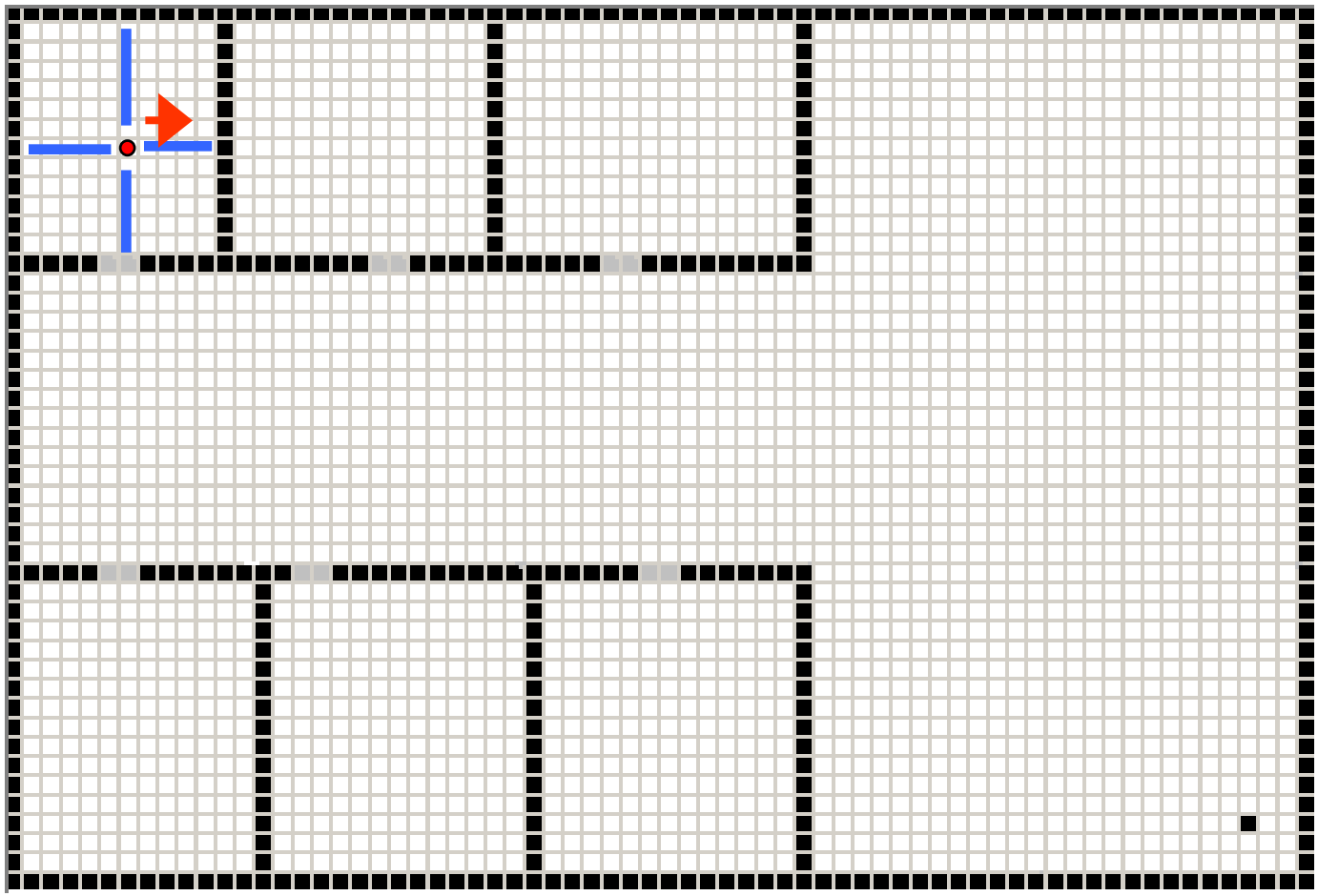
- Intuitive test problem; HASSLE is not constrained to navigation or to specific environments (MDPs and POMDPs)
- Objective: move to the goal position from any random start position
- 10,961 states
- Reward r=4 when agent reaches the goal, r=0 otherwise: strongly delayed reward

Low-level actions and observations

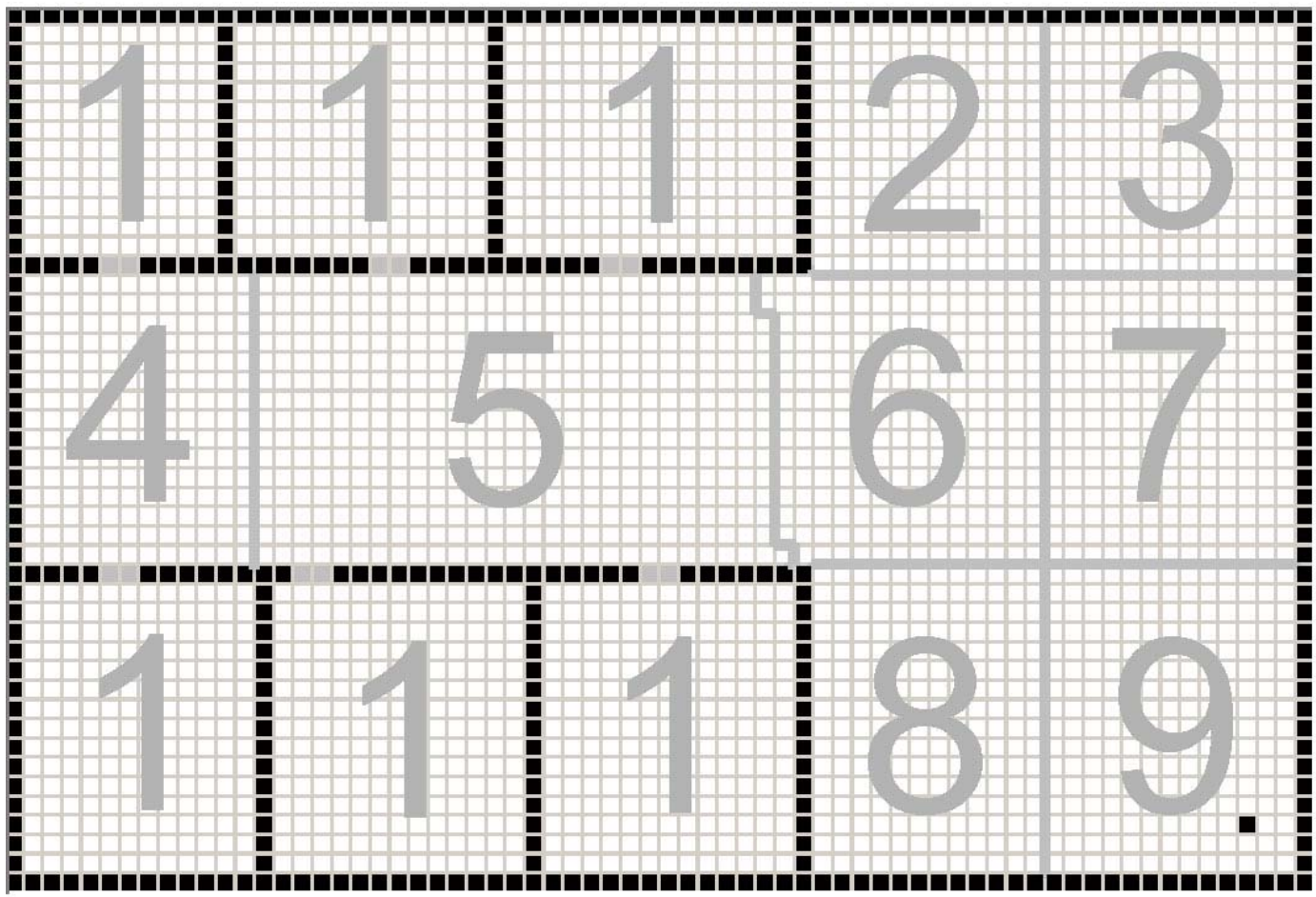
- primitive (low-level) actions
  - move in the current direction
  - turn left 90º
  - turn right 90º
- primitive (low-level) observations
  - 4 sensors that detect doors
  - 4 sensors that detect goal (if within 8 grid cells)
  - agent's orientation
  - (o^H_S, o^H_G)

used for **high-level** observations as well:
unsupervised classification of this 4D vector

Office low-level sensors



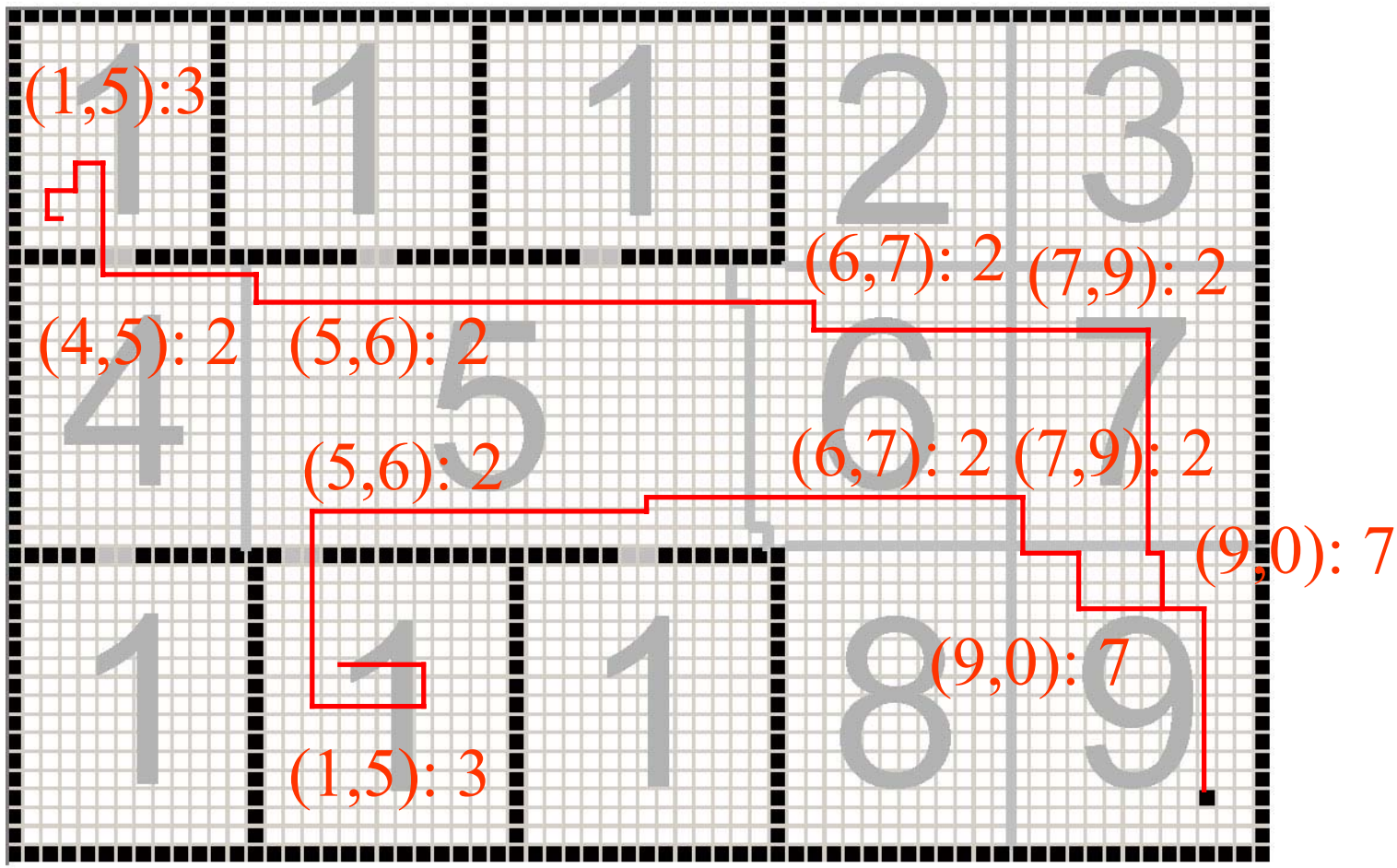
Office high-level observations



Policy implementation

- **High-level policy**
  - table-based
  - inputs: high-level observations
  - outputs: values of subgoals (high-level observations)
- **Low-level policies**
  - 8 low-level policies
  - linear function approximators (perceptrons)
  - inputs: low-level observation vectors
  - outputs: values of low-level, primitive actions

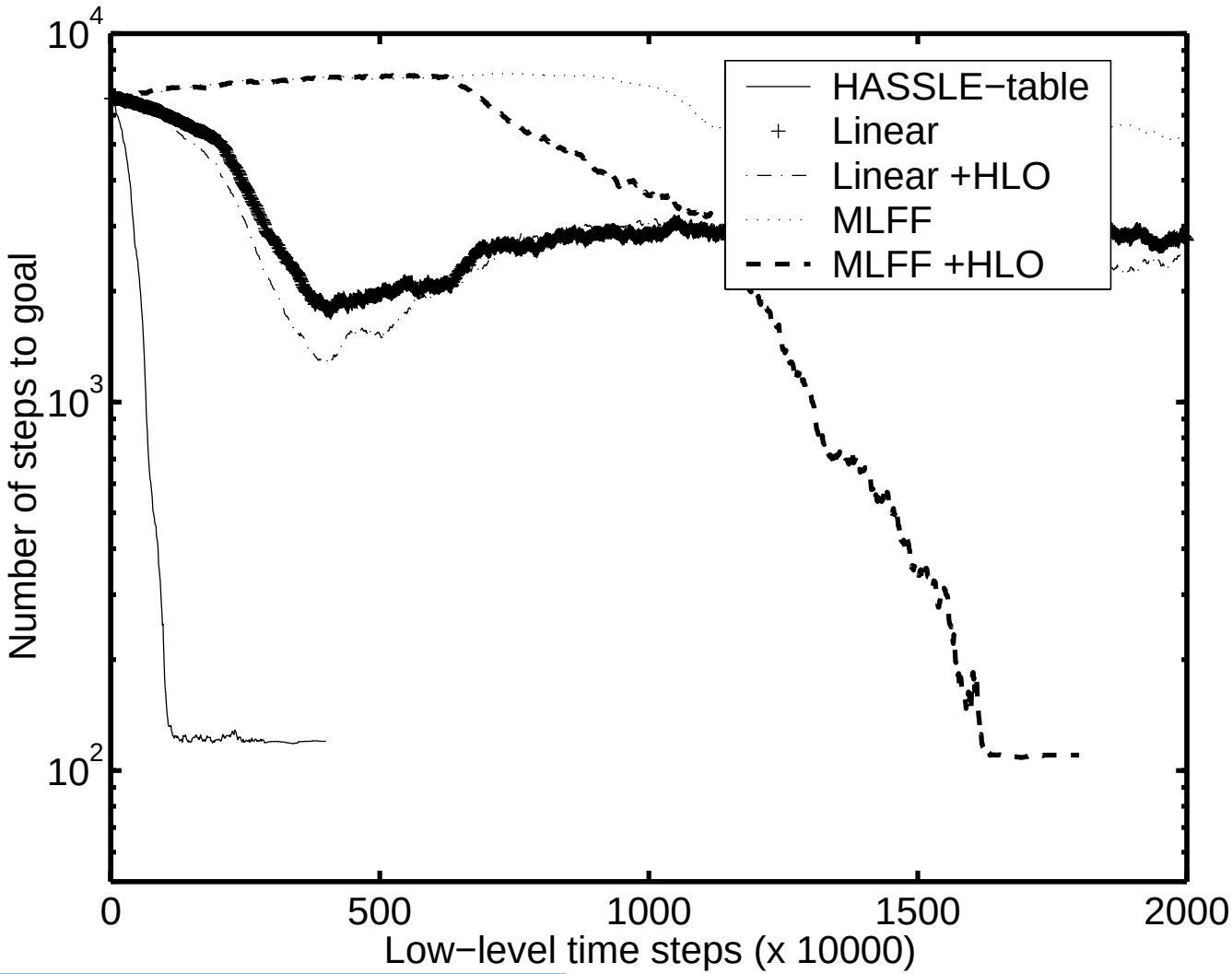
Example trajectories after learning



Comparison with **flat** RL methods

- linear function approximator
- nonlinear function approximator (multilayer feedforward neural network – MLFF)
- linear function approximator + *high-level observation as extra input*
- nonlinear function approximator + *high-level observation as extra input*

Learning results



Other experiments

- Same office task, but with noise on low-level observations, high-level observations, and low-level actions: *similar results*
- Same office task, but with different numbers of low-level policies: *similar results*
- Similar office task, but with partial observability at higher level: *similar results but slower*
- Similar office task, but with 3 levels rather than 2: *promising initial results*

Conclusions: virtues of HASSLE

- HASSLE's initial results are promising
  - it learns useful decompositions of tasks
  - it learns useful specialization and generalization by subpolicies
  - it learns faster than several flat RL methods
- HASSLE is general
  - various function approximators are possible (also for POMDPs!)
  - various value function RL algorithms are possible
  - various clustering algorithms are possible to produce the high-level observations
  - more than 2 levels are possible

Conclusions: limitations of HASSLE

- Many tunable parameters
- Convergence, even to local optimum, will be difficult to prove
- Dependence on reasonable clustering of low-level observations into high-level observations

Future work

- Work on limitations
- Adapting high-level observations online so as to improve their utility as subgoals
- Application to a real robot task