Decision Making for Robots and Autonomous Systems

Fall 2015

DUKE COMPUTER SCIENCE

George Konidaris
gdk@cs.duke.edu

Hierarchical RL

RL typically solves a *single* problem *monolithically*.

Hierarchical RL:

- Create and use higher-level macro-actions.
- Problem now contains subproblems.
- Each subproblem is also an RL problem.

Options Framework: theoretical basis for skill acquisition, learning and planning using higher-level actions (options).

The Options Framework

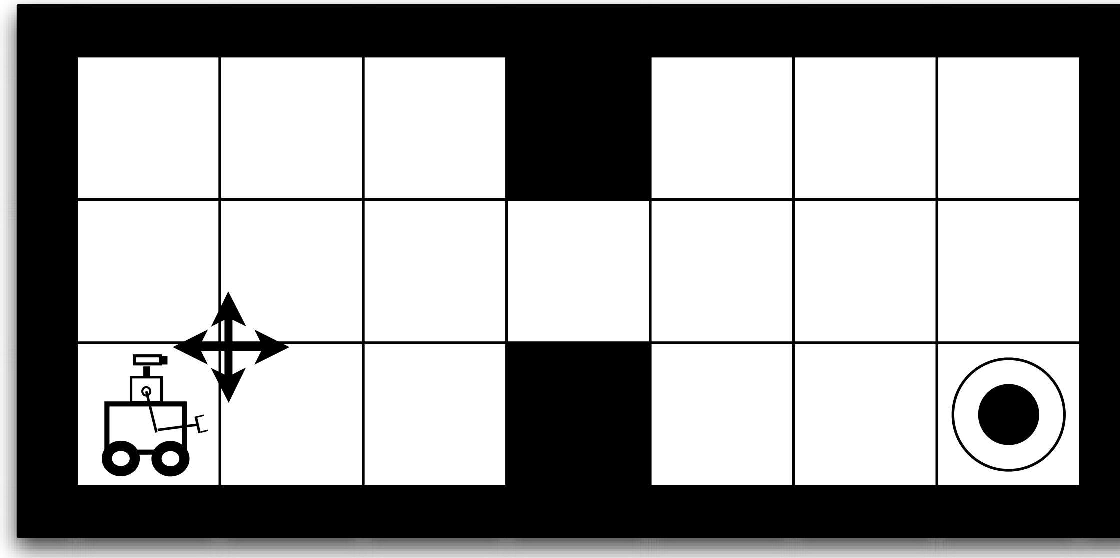
Basic idea:

- Define a *temporally extended action* as a *policy*.

A (Markov) option *o* is a policy unit:

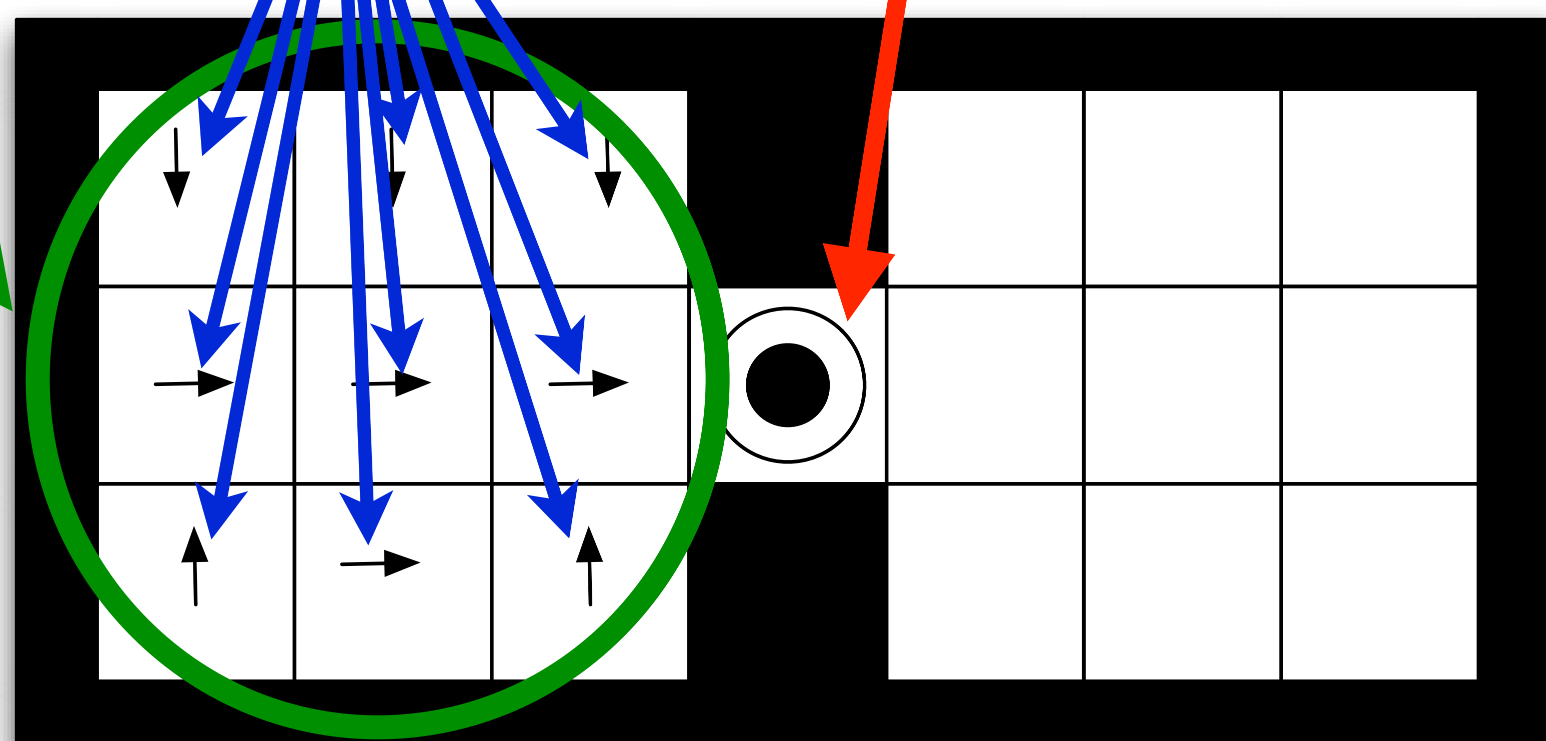
- Initiation set $I_o : S \to \{0, 1\}$
- A termination probability $\beta_o : S \to [0, 1]$
- A policy $\pi_o : S \times A \to [0, 1]$

More Intuitively



An option *o* is a policy unit:

- Initiation set
- Termination condition
- Option policy

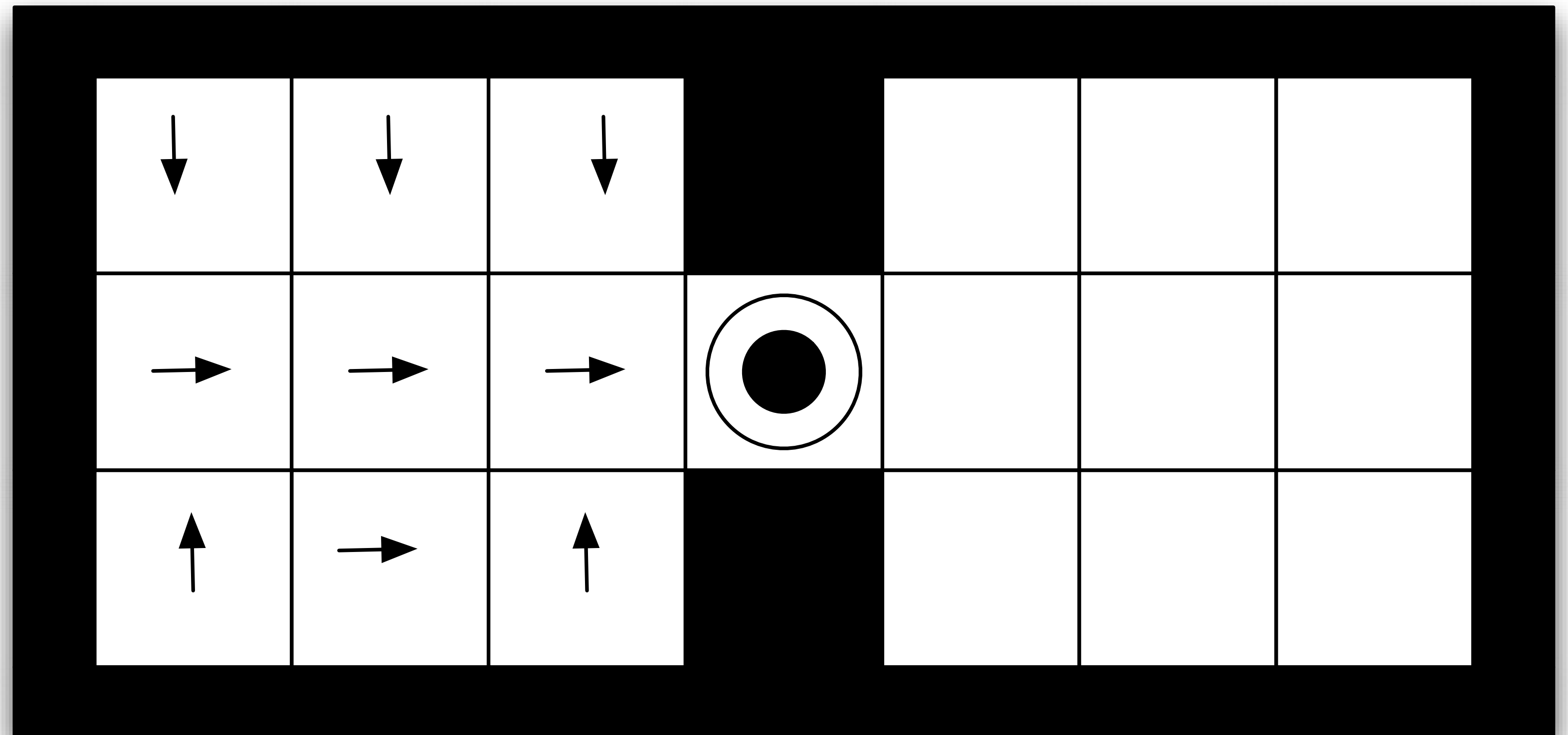


Notes

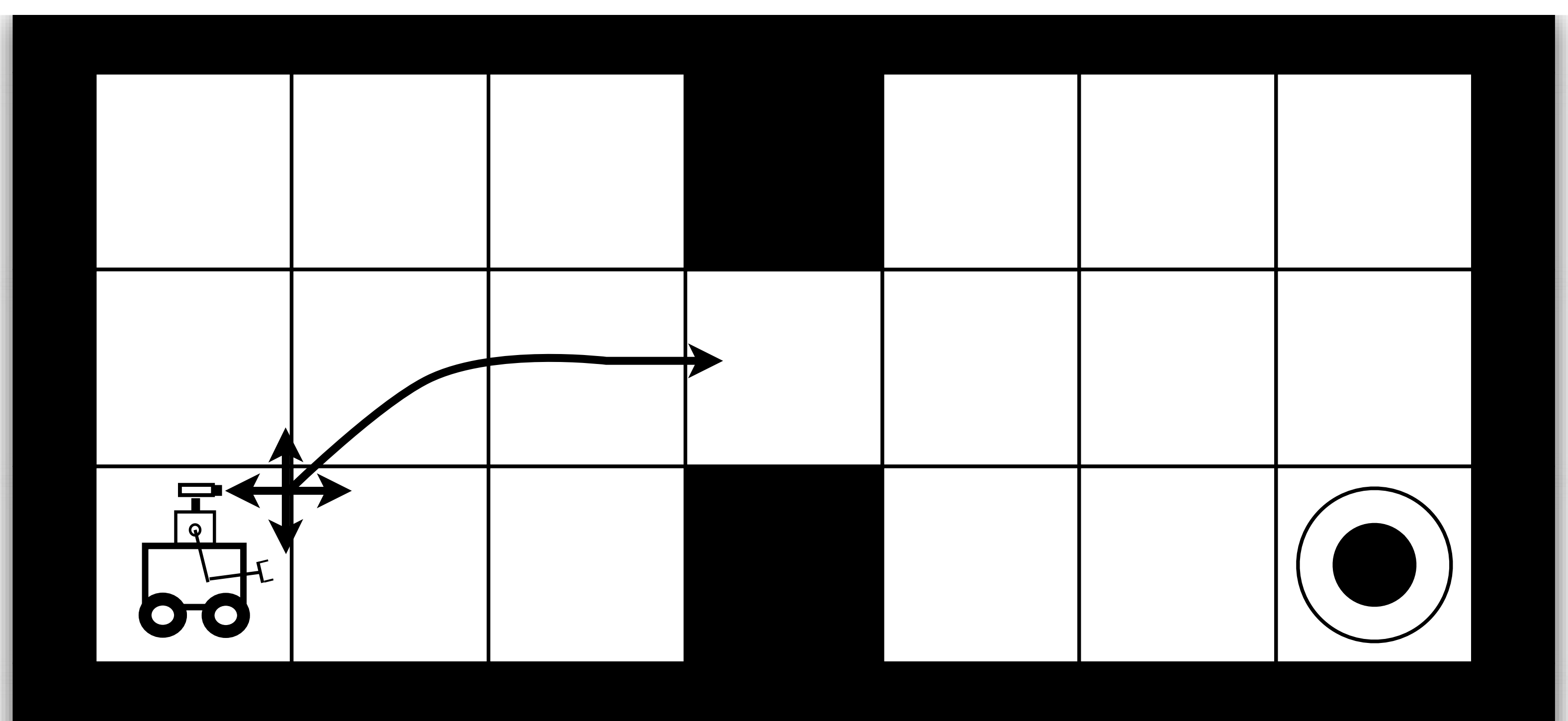
- Given R_o, learning π_o is just another (episodic) RL problem.
- Typically only need to define π_o over I_o.
- Equally, π_o could be any policy (generically, a program).

Options as Actions

Option



Problem



SMDPs

The resulting problem is a *Semi-(Markov Decision Process)*.
This consists of:

- S Set of states
- O Set of options
- $P(s', t|o, s)$ Transition model
- $R(s', s, t)$ Reward function
- γ Discount factor (per step)

In this case:

- All times are integers.
- "Semi" here means transitions can last *t* timesteps.
- Transition and reward function involve time taken for option to execute.

So:

Original problem: MDP.
MDP + Options = SMDP.

Options framework allows us to both *express a low-level policy*, and *plan and learn using the higher-level SMDP*.

Additionally, the ability to:

- Create new options.
- Update option policies.
- Do off-policy learning using, or for, them.
- Interrupt them ...

puts us "between MDPs and semi-MDPs".

What are Skills For?

Lots of things!

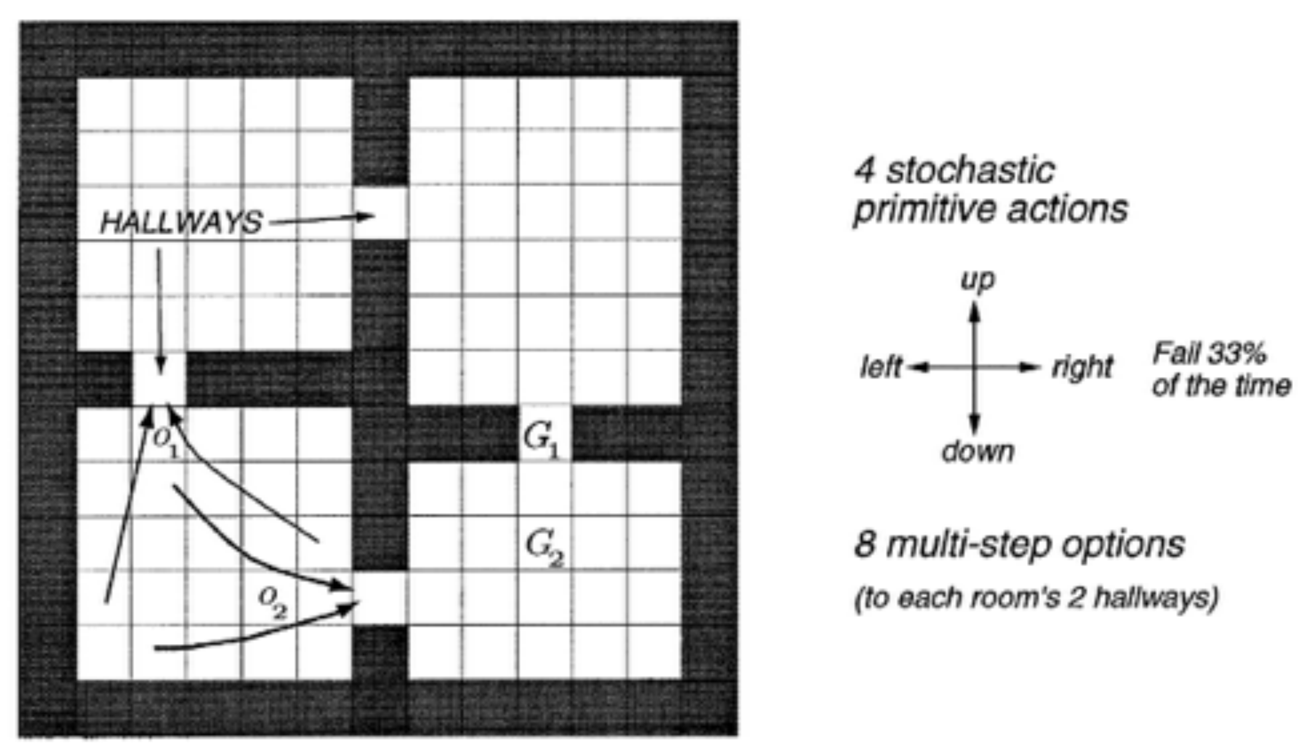
A few salient points:

- Rewiring.
- Transfer.
- Skill-Specific Abstractions.

Rewiring

Adding an option changes the connectivity of the MDP.
This affects:

- Learning and Planning.
- Exploration.
- State-visit distribution.
- *Diameter of problem.*



(Sutton, Precup and Singh, AIJ 1999)

Transfer

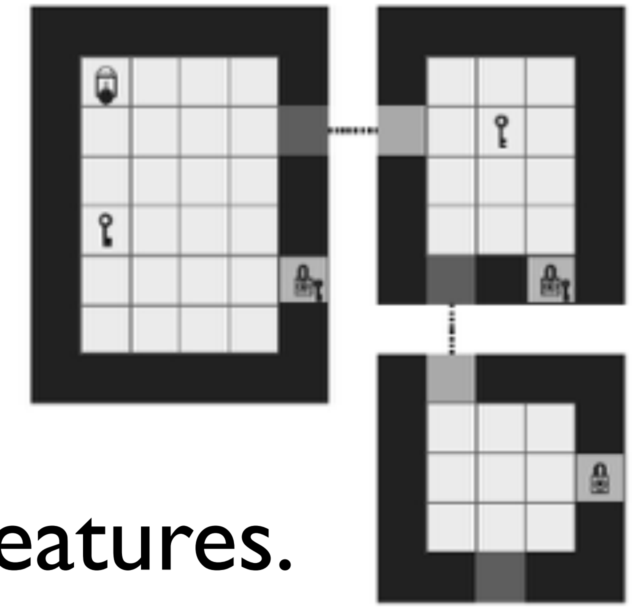
Use experience gained while solving one problem to improve performance in another.

Skill transfer:

- Use options as mechanism for transfer.
- Transfer *components* of solution.
- Can drastically improve performance
- ... even if it takes a lot of effort to learn them.

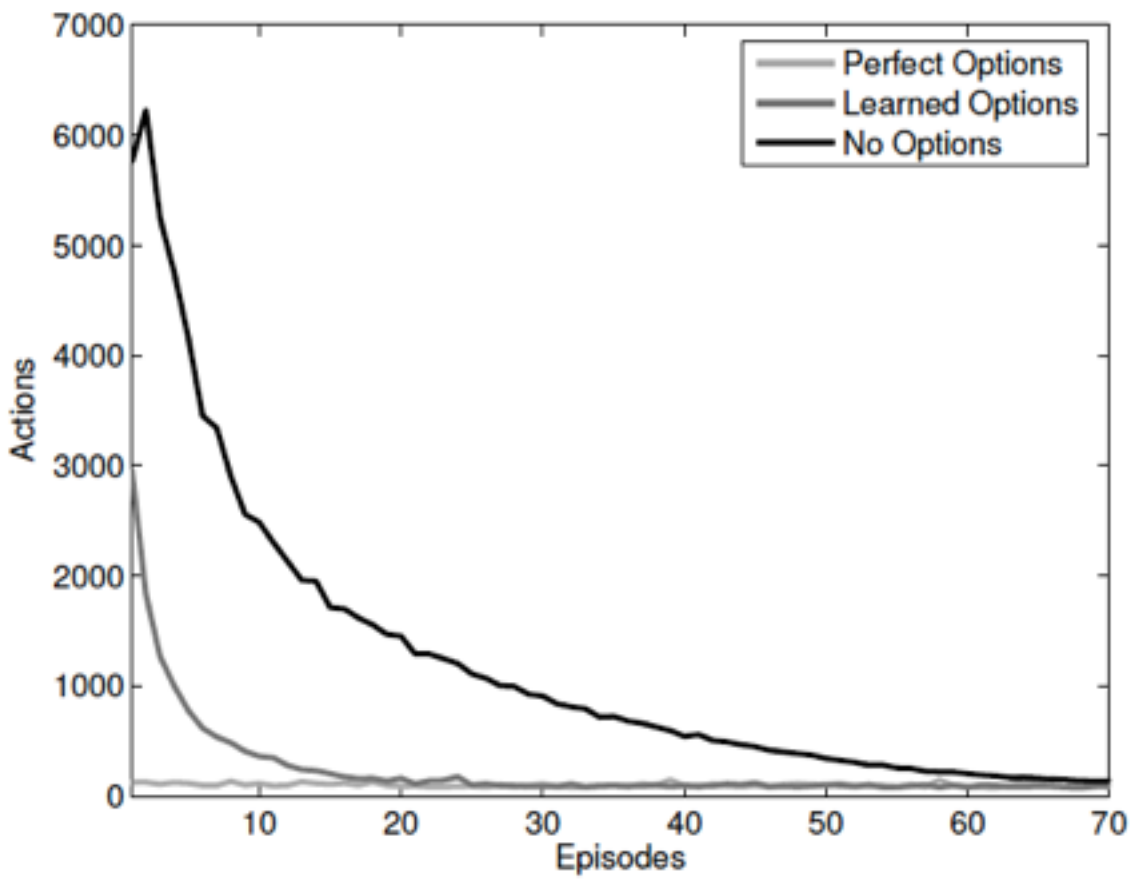
General principle: **subtasks recur.**

Example

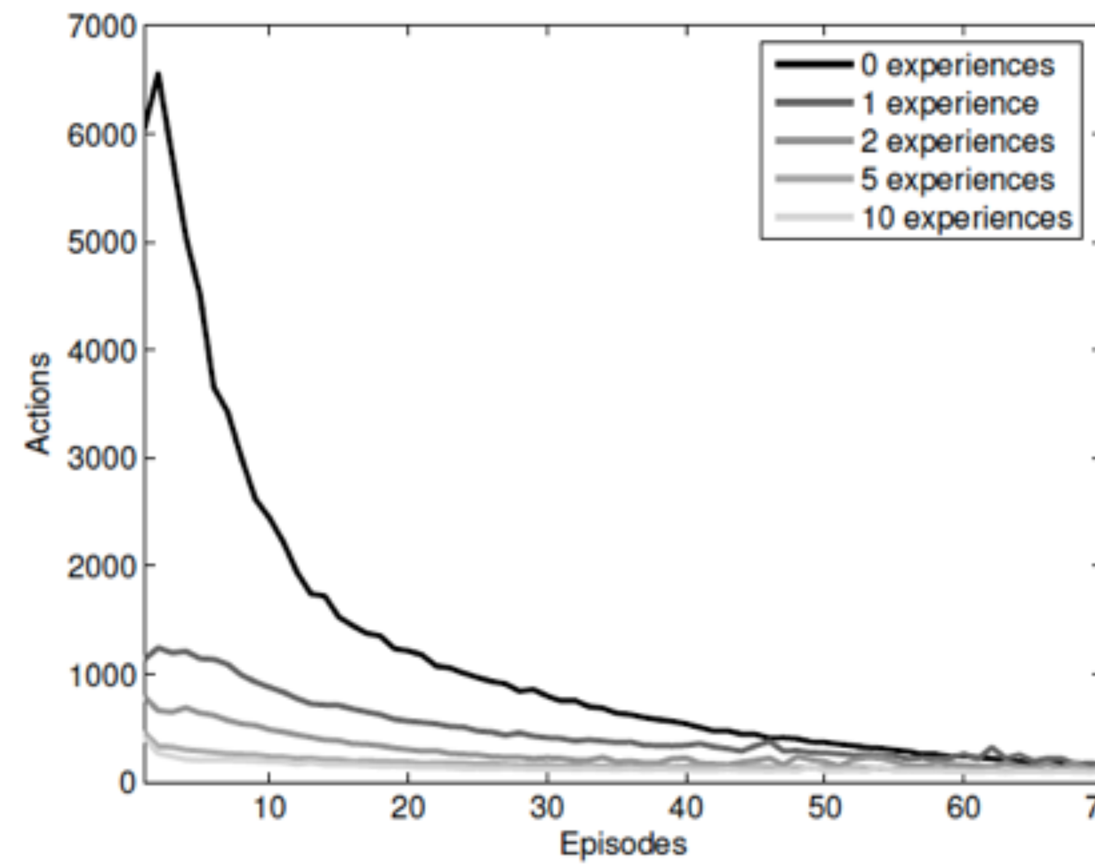


Tasks drawn from parametrized family.

- Common features present.
- Options defined using only common features.



(a) Learning curves for agents with problem-space options.



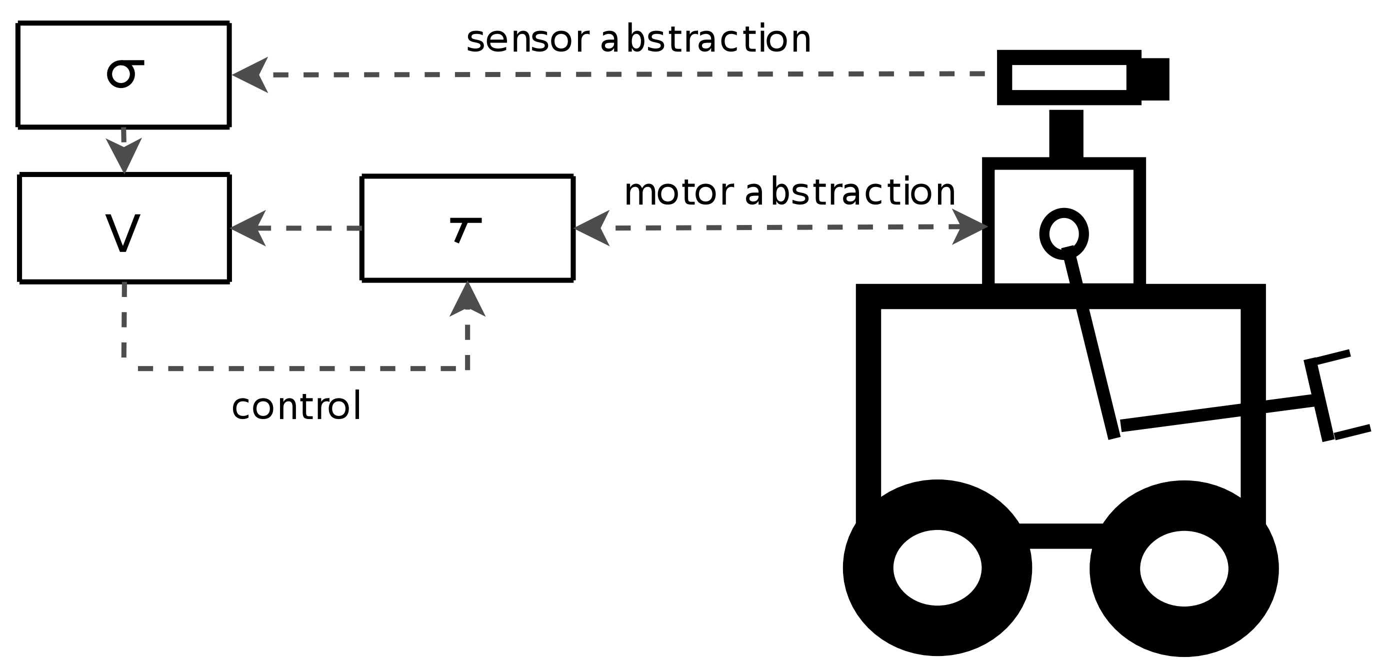
(b) Learning curves for agents with agent-space options, with varying numbers of training experiences.

(Konidaris and Barto, IJCAI 2007)

Skill-Specific Abstractions

Common approach to solving hard problems:

- Use an abstraction!



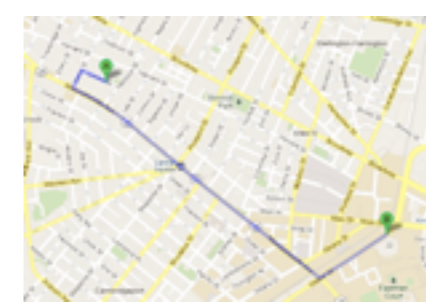
But

- Many high-dimensional problems *really are* high-dimensional *if you try to solve them monolithically*

Skill-Specific Abstractions

Options provide an alternative approach:

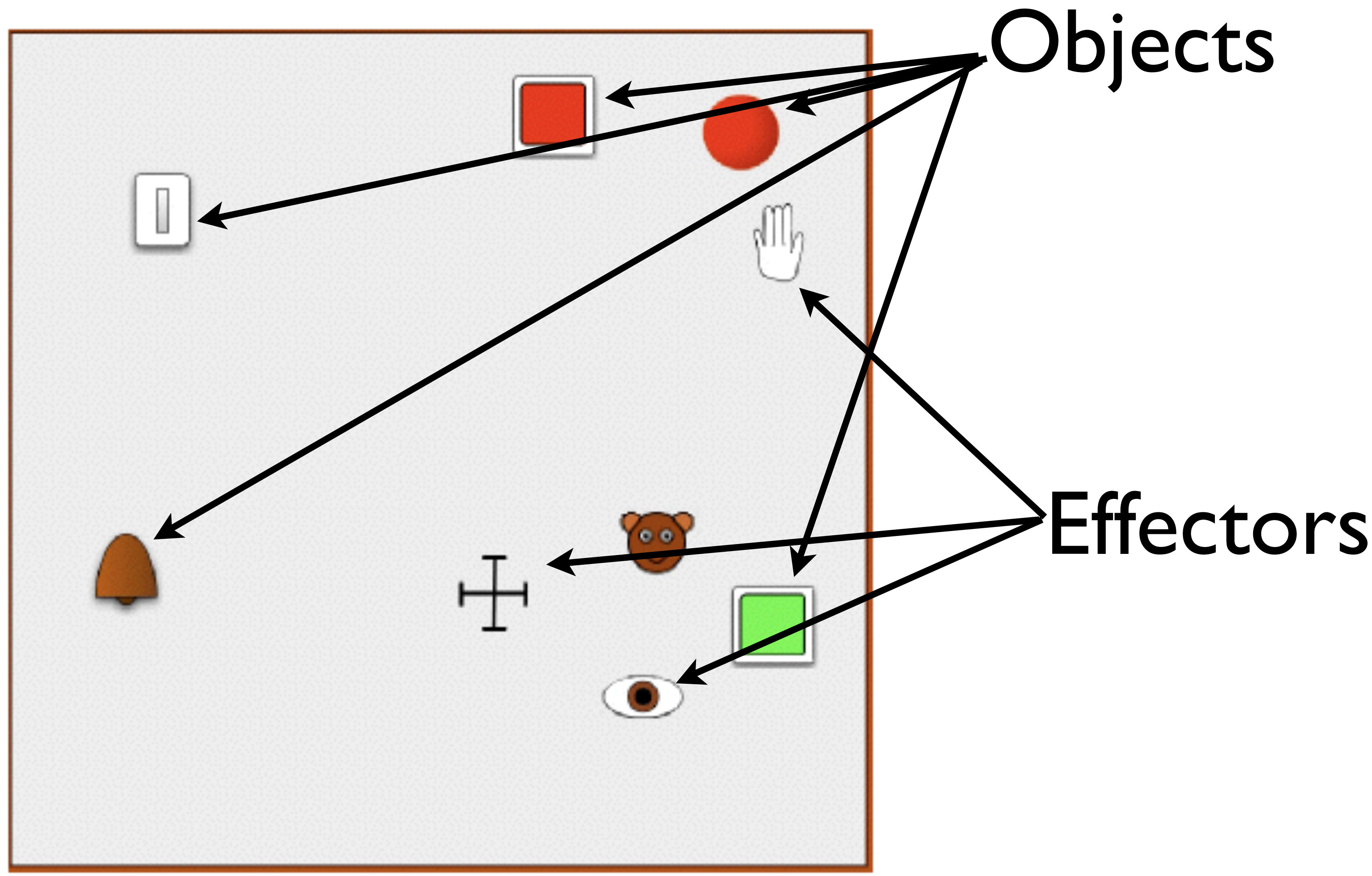
- Split high-dimensional problem into subproblems …
- … such that each one supports a solution using an abstraction.



Working hypothesis: *behavior is piecewise low-dimensional.*

The Continuous Playroom

The Continuous Playroom

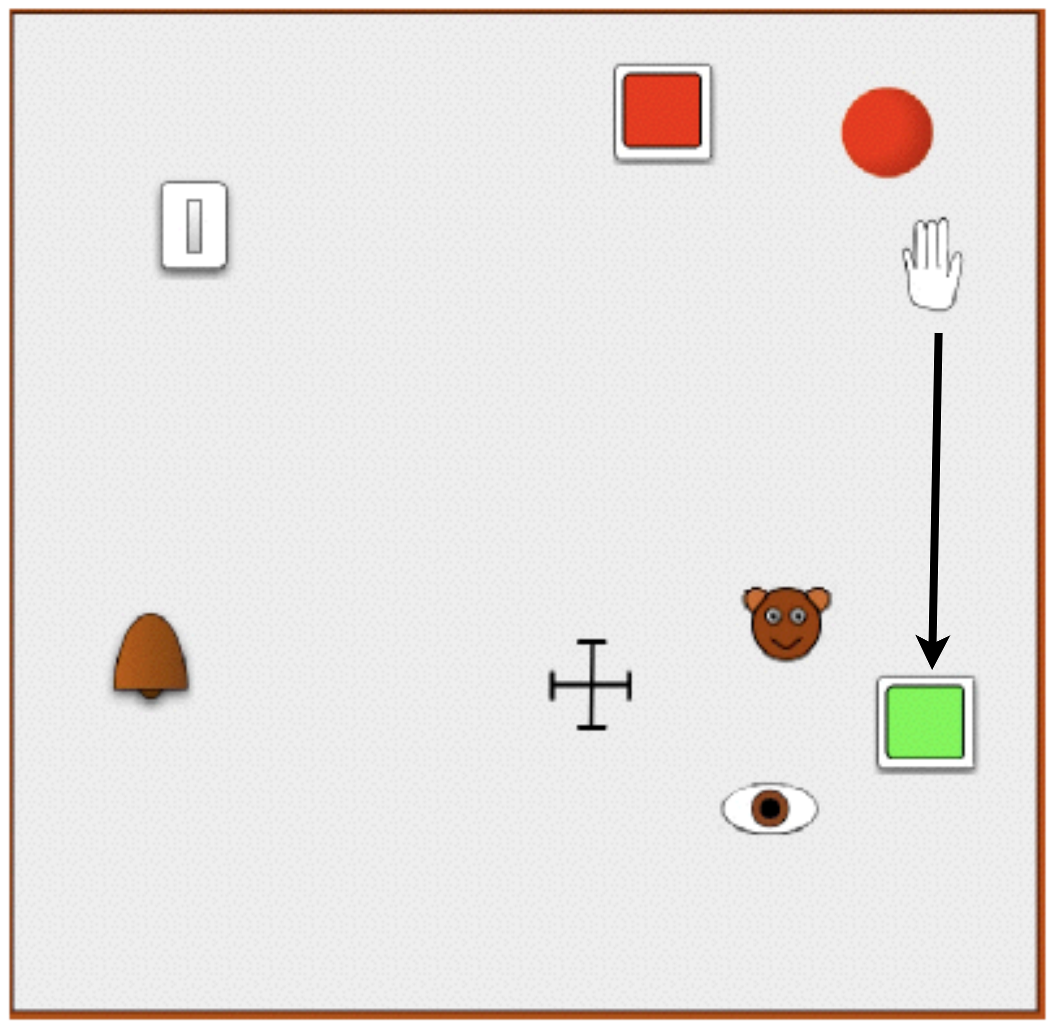


Randomly re-arranged between episodes.
120 state features.

(Konidaris and Barto, IJCAI 2009)

The Continuous Playroom

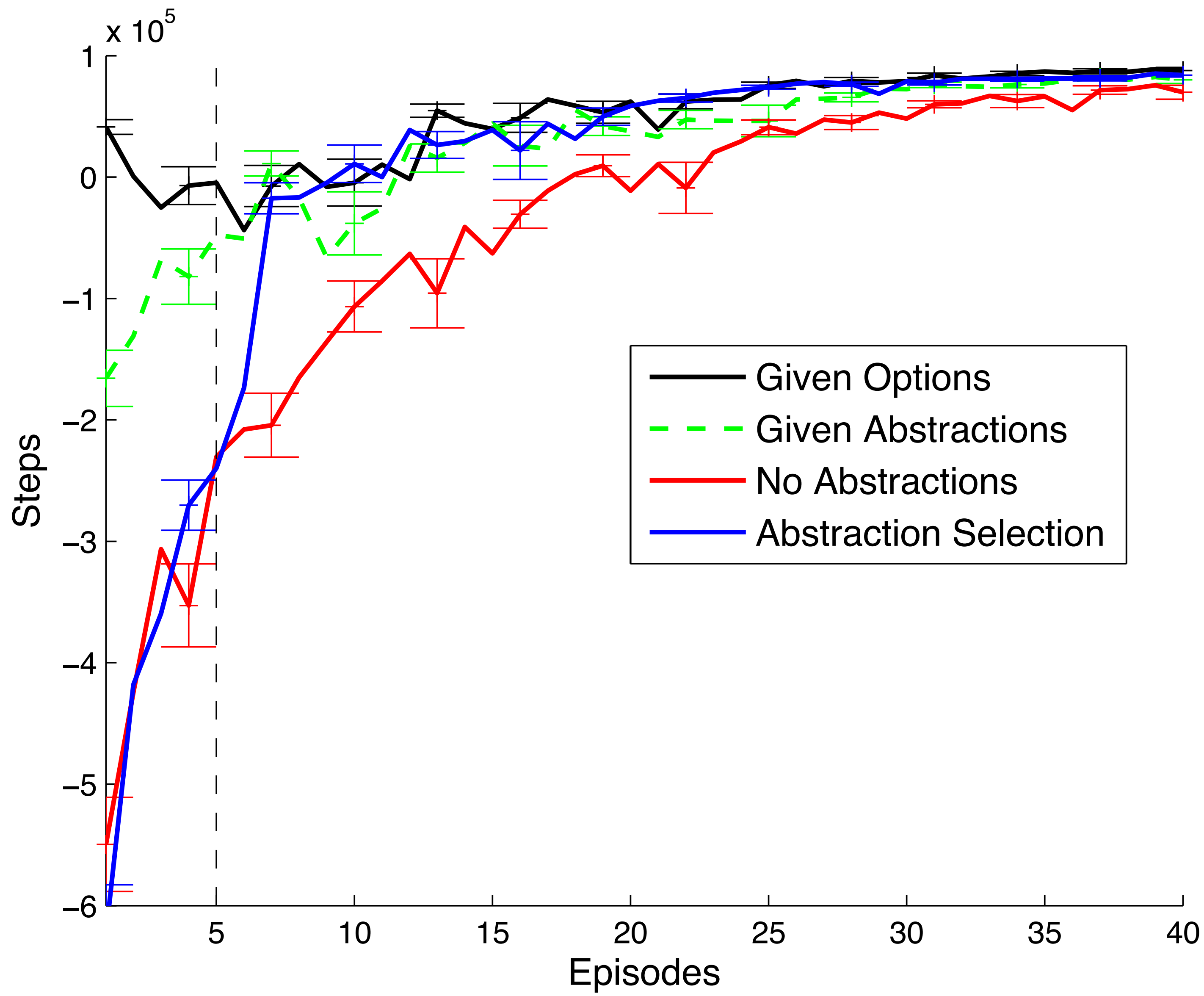
Skills: *placing each effector over an object (allow interaction)*



Available abstractions:

- *x* and *y* differences for each object-effector pair.

Experiments



Skill Discovery

Discover options autonomously, through interaction with an environment.

- Typically *subgoal options*.
- This means that we must determine β_o.
- Sometimes also R_o.

The question then becomes:

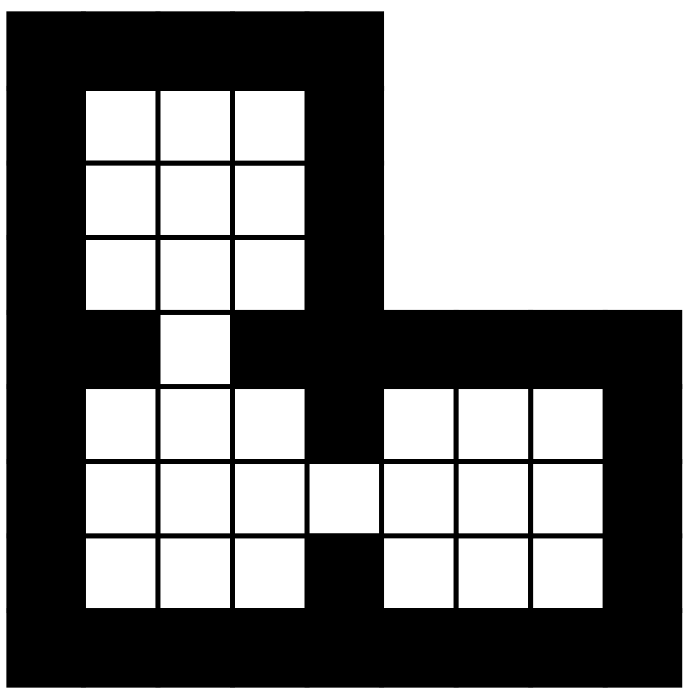
- Which states are good subgoals?

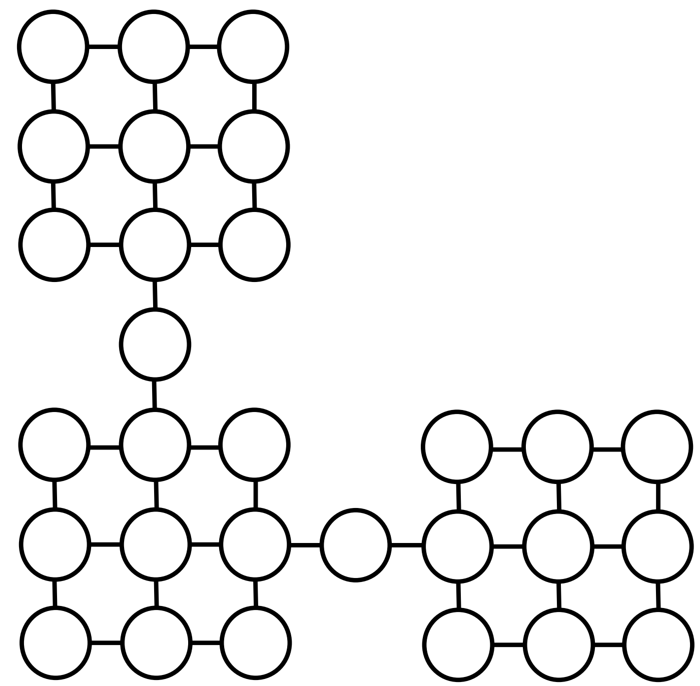
There are several ways to answer this.

Betweenness Centrality

Consider an MDP as a graph.

- States are vertices.
- Edges indicate possible transition between two states.





Further, let us assume a task distribution over start states and goal pairs:

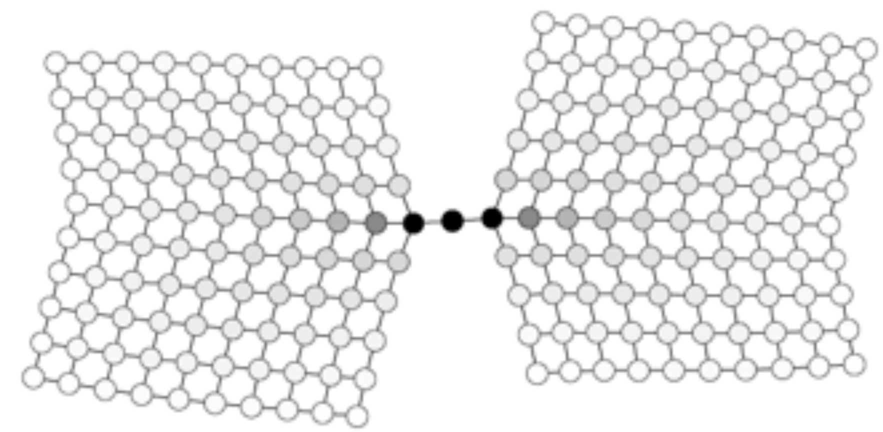
- $P_T(s, e)$

(Simsek and Barto, NIPS 2008)

Betweenness Centrality

We can define the *betweenness centrality* of a vertex (state) as:

$$\sum_{s,e} \frac{\sigma_{se}(v)}{\sigma_{se}} w_{se}$$



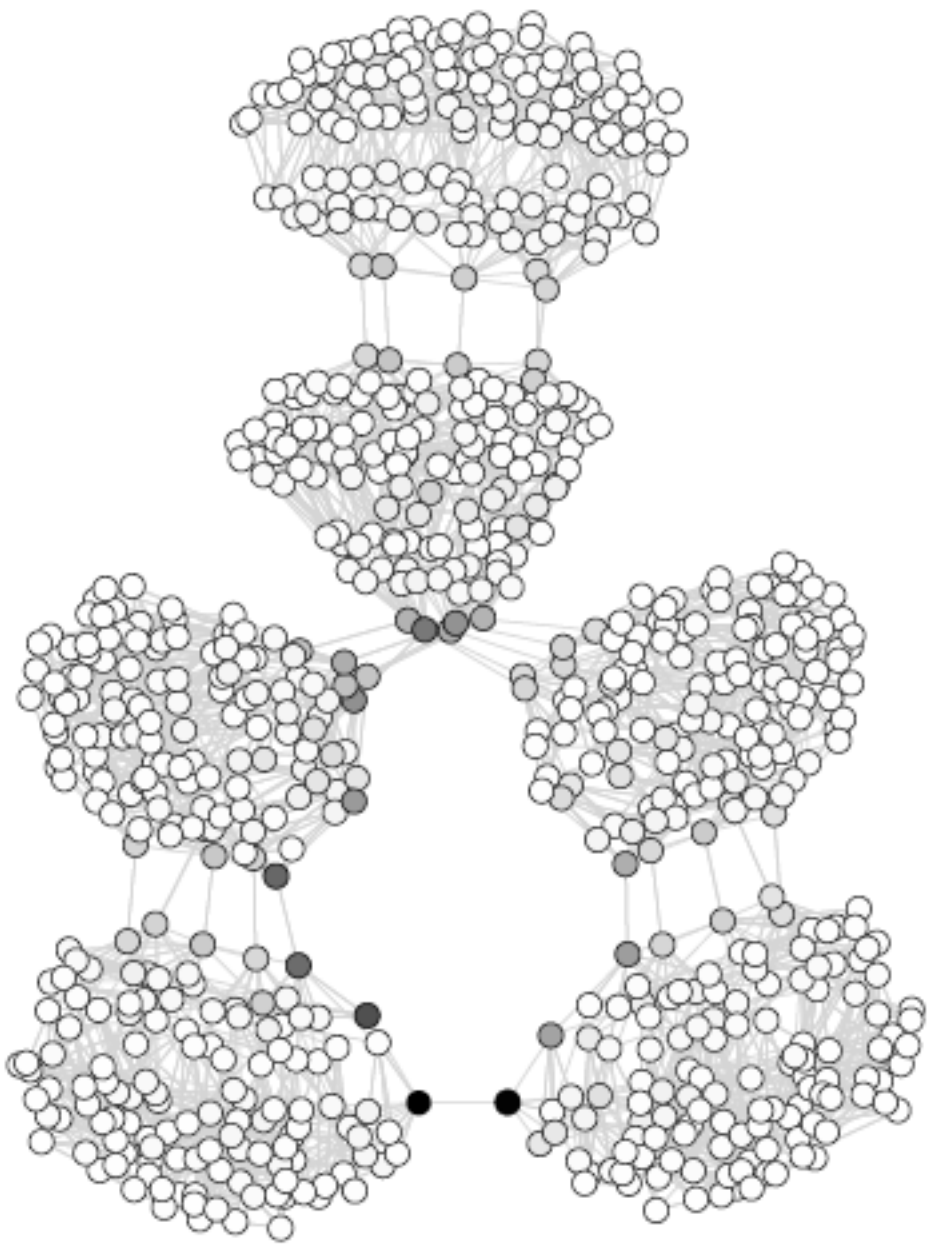
This indicates it probability of being on a shortest path from *s* to *e*; if we define:

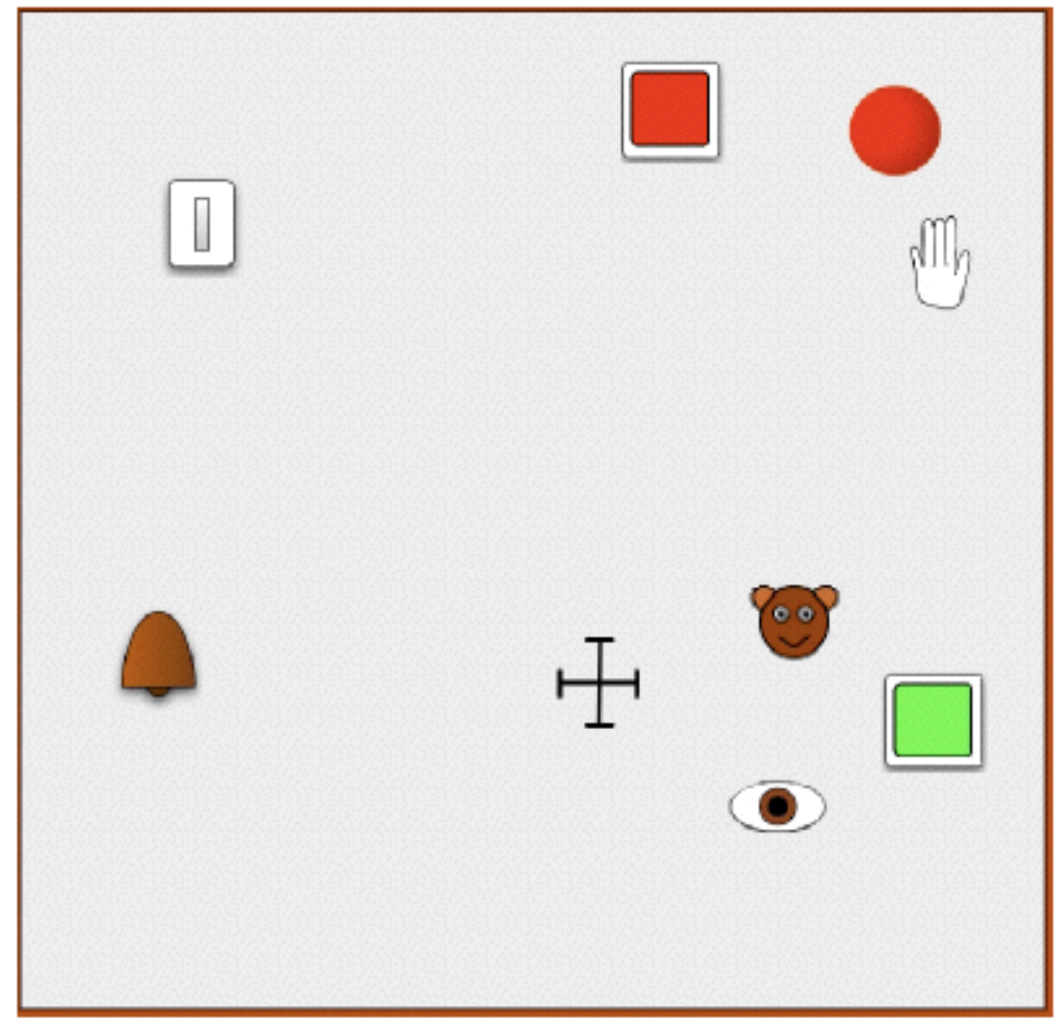
- *Shortest path* as *optimal solution*.
- $w_{se} = P_T(s, e)$

... then we get something sensible for RL.

(Simsek and Barto, NIPS 2008)

Betwenness Centrality





(Simsek and Barto, NIPS 2008)

Betweenness Centrality

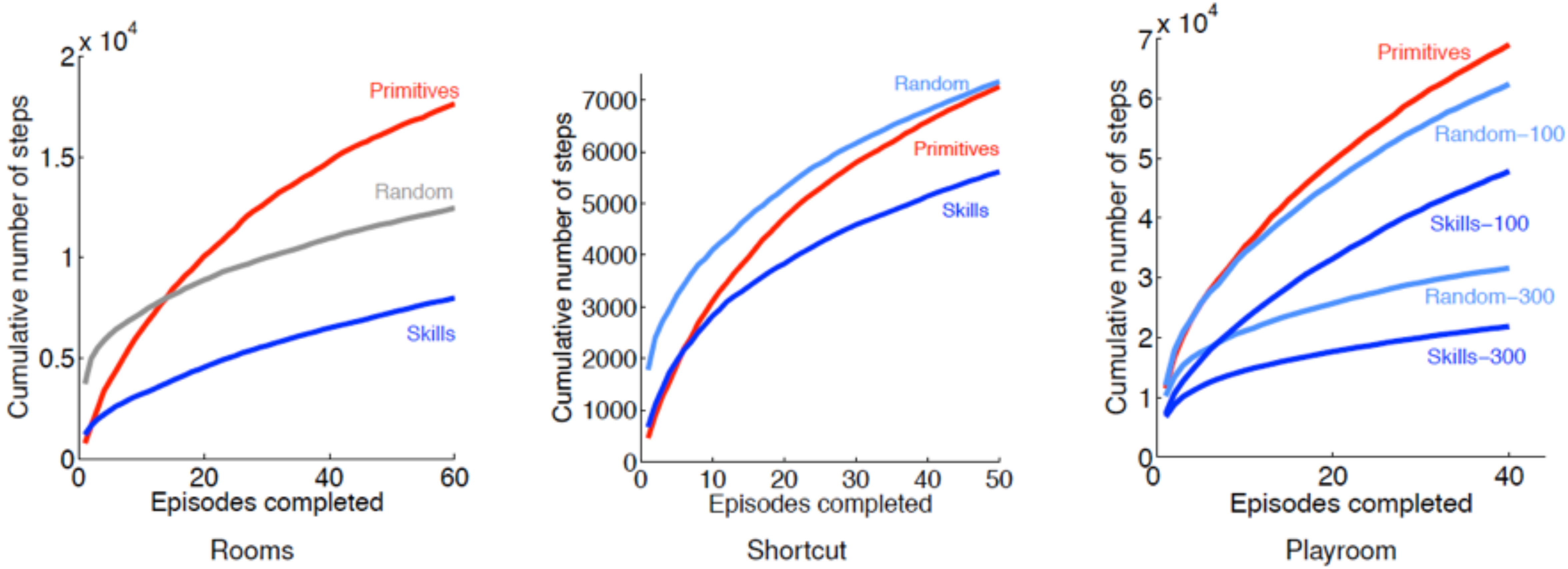


Figure 3: Learning performance in Rooms, Shortcut, and Playroom.

(Simsek and Barto, NIPS 2008)

Betweenness Centrality

Of course:

- Knowing the MDP is cheating.
- So is knowing the distribution of problems.
- But can use this as the basis for approximation.

Continuous State Spaces

Continuous state spaces are more challenging:

- Need a goal region, not a state.
- Cannot assume $I_o = S$

For episodic tasks:

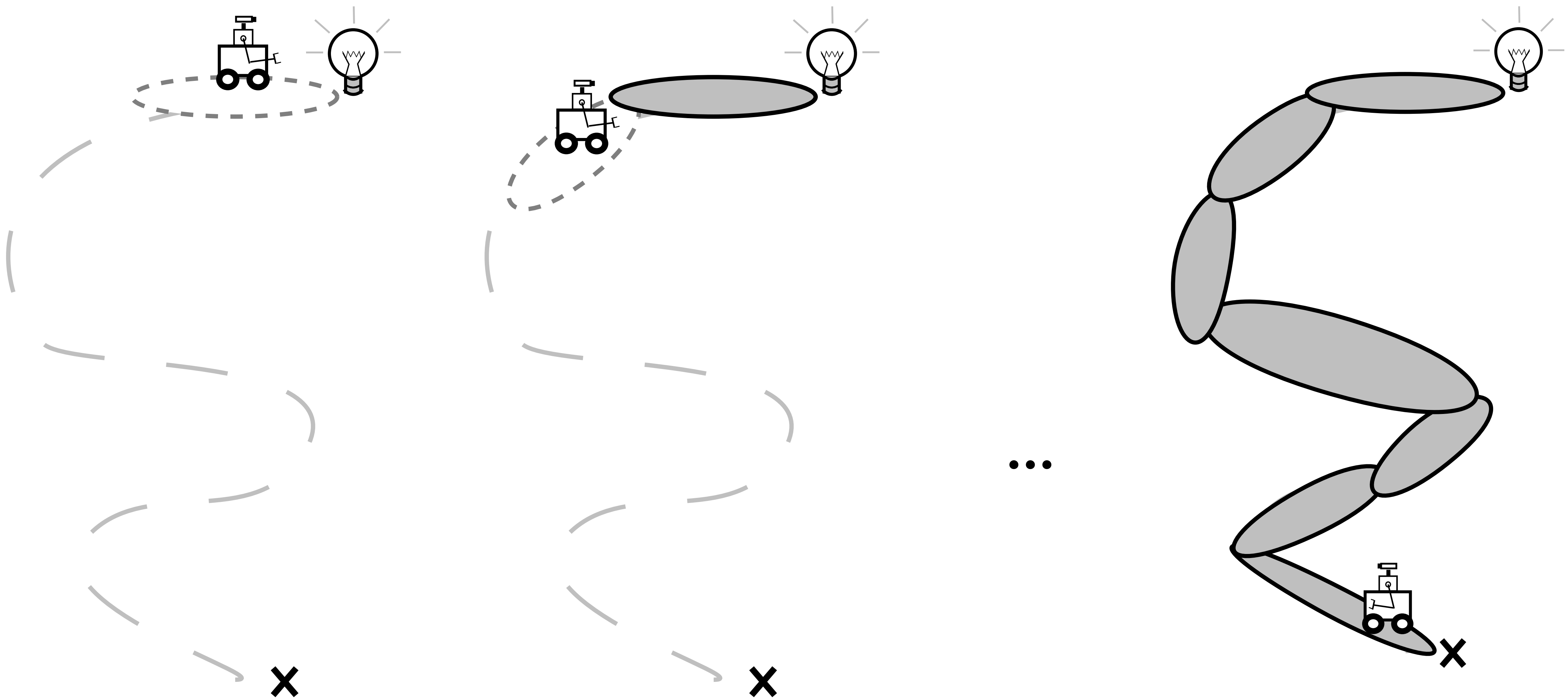
- End-of-episode is a good target.
- Can we generate more?

The point of executing a skill is either to:

- Get to a solution
- Get to another skill that might lead to a solution

Skill Chaining

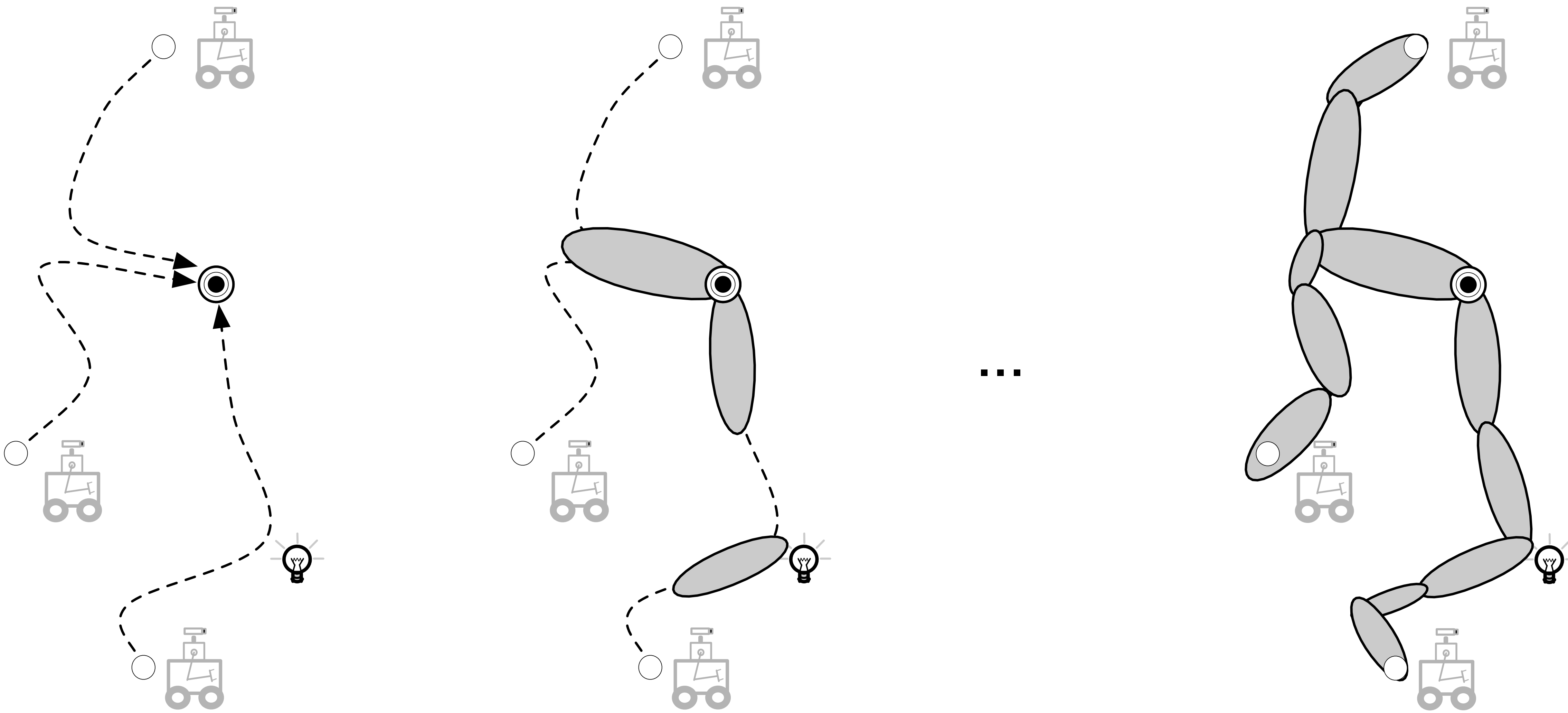
Simple rule: *when creating a new skill to reach a target event, make entering that skill's initiation set a new target event.*



(Konidaris and Barto, NIPS 2009)

Skill Chaining

Problems are not usually that clean.



Skill Chaining

Skill goal is a region, not a state.

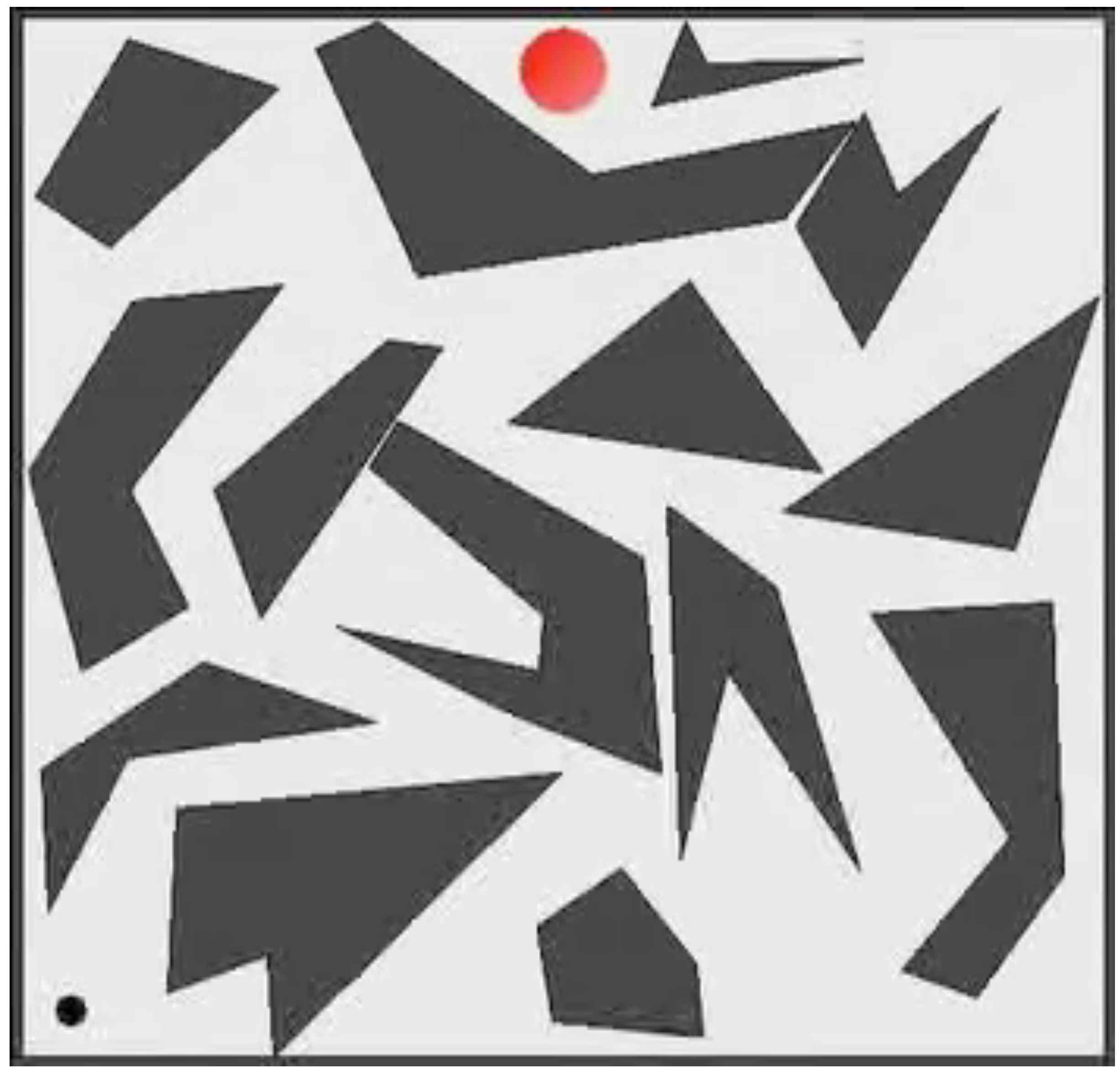
Initiation set learned using classifier.

- Execute and fail: −
- Execute and succeed: +

Can include other target events:

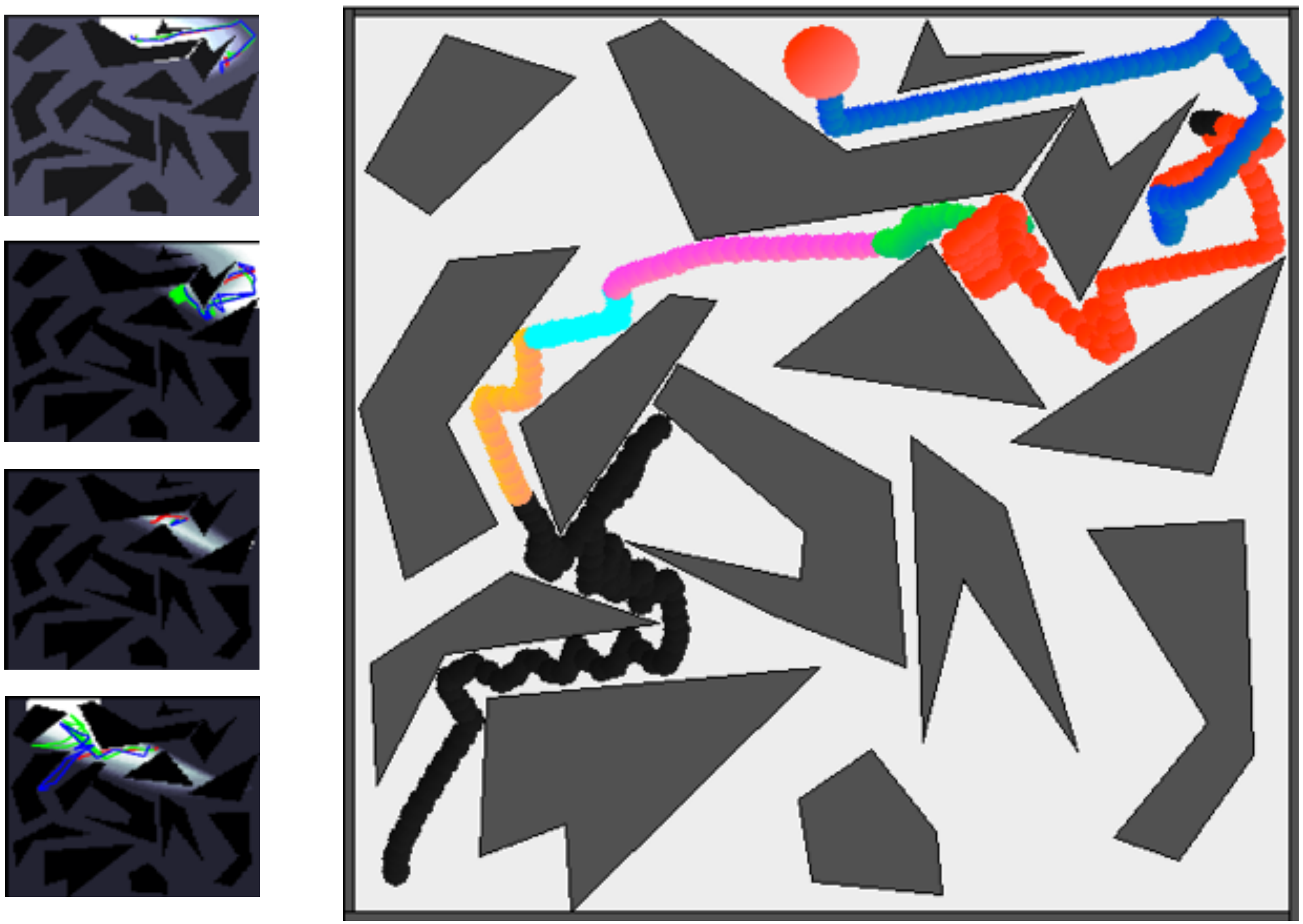
- Domain knowledge
- Other heuristics
- *Still need to chain to overcome limited range*

Skill Chaining



(Konidaris and Barto, NIPS 2009)

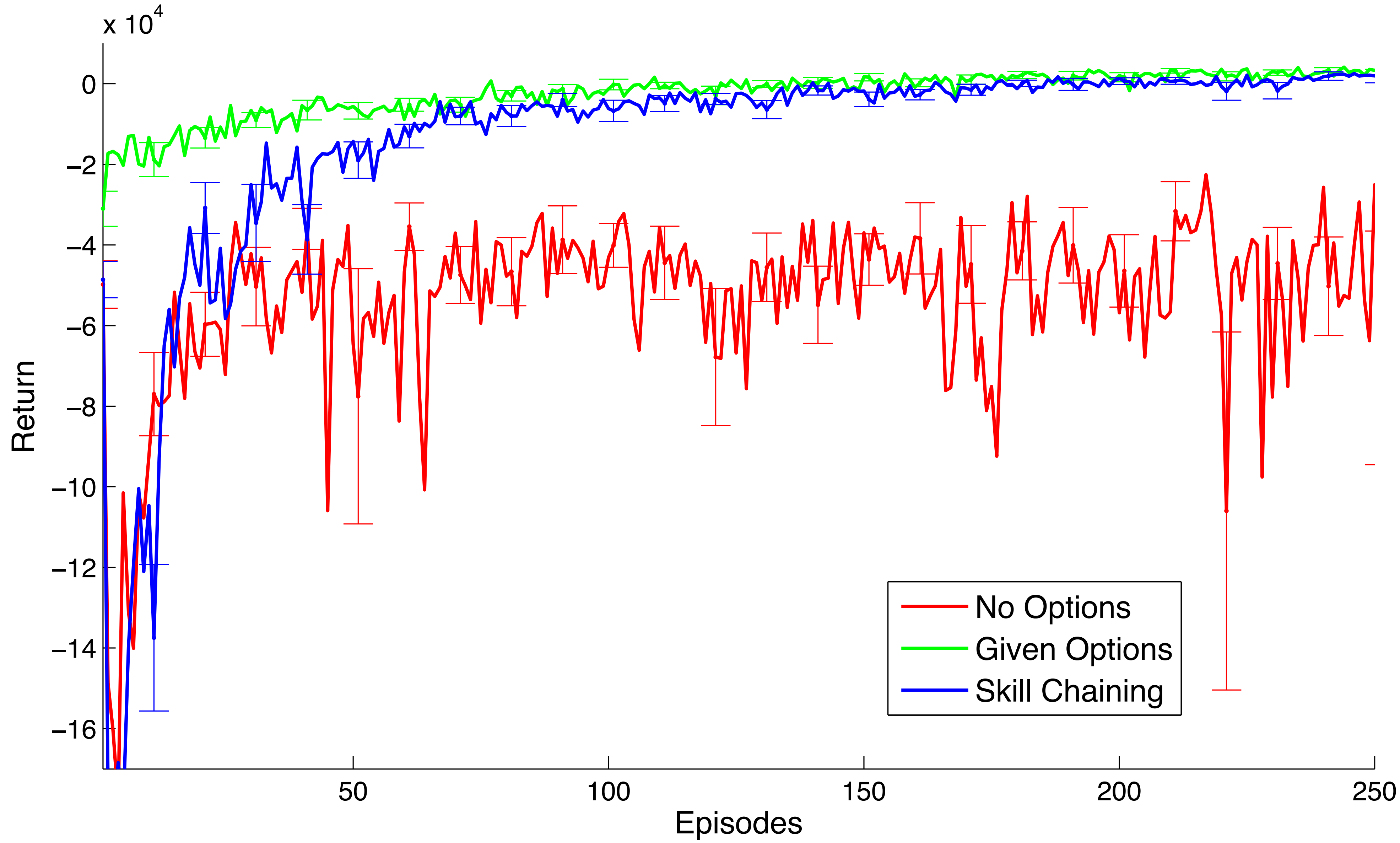
Skill Chaining



(Konidaris and Barto, NIPS 2009)

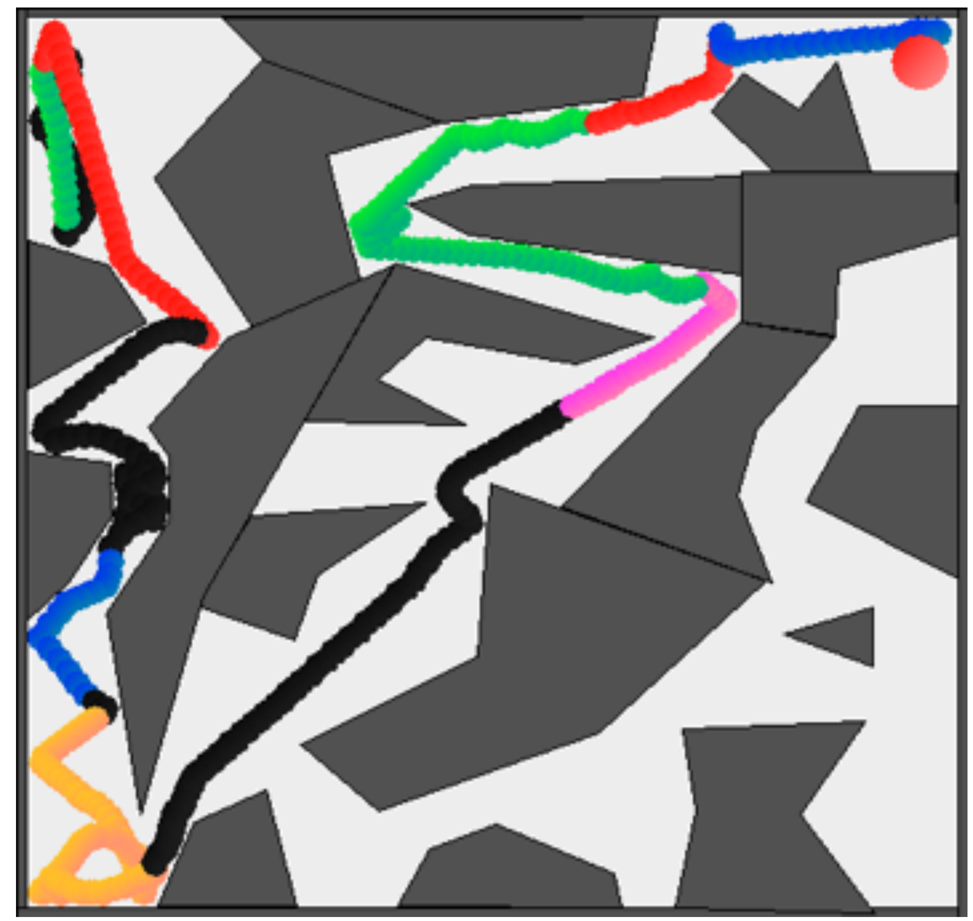
Skill Chaining

DUKE COMPUTER SCIENCE



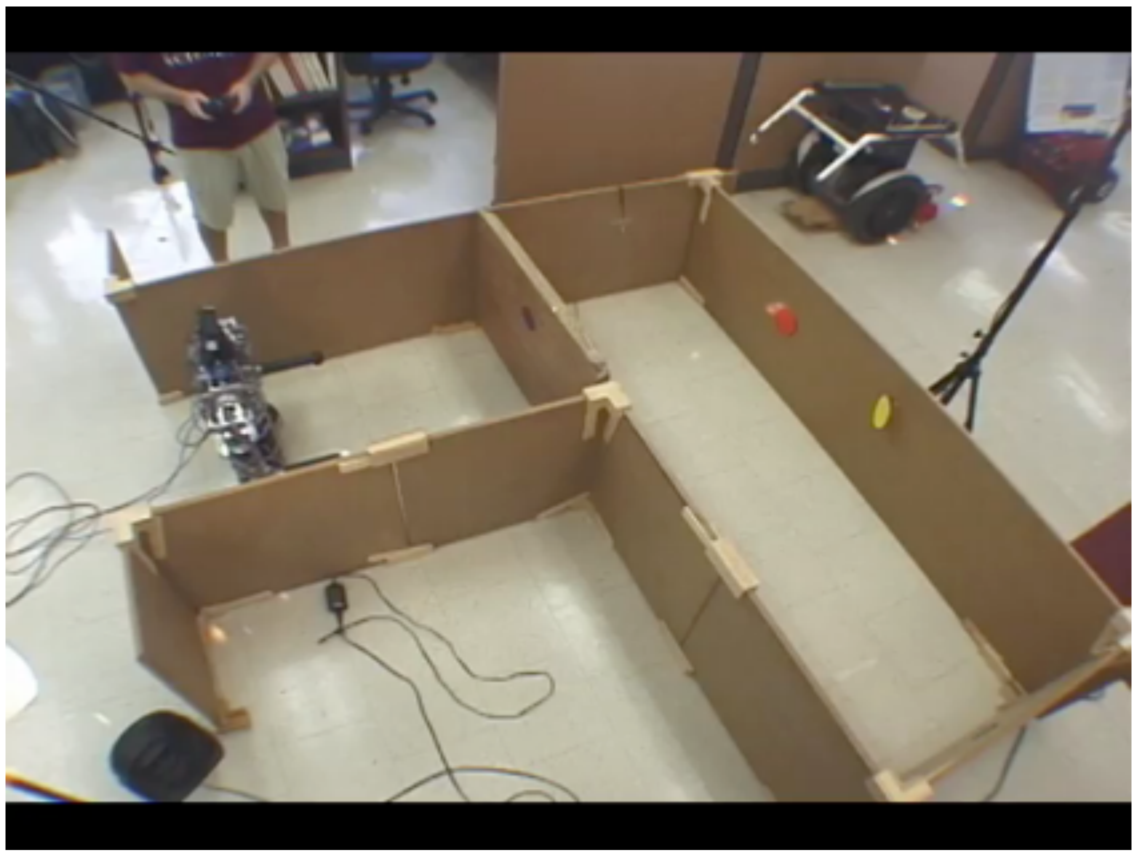
Skill Chaining: Results

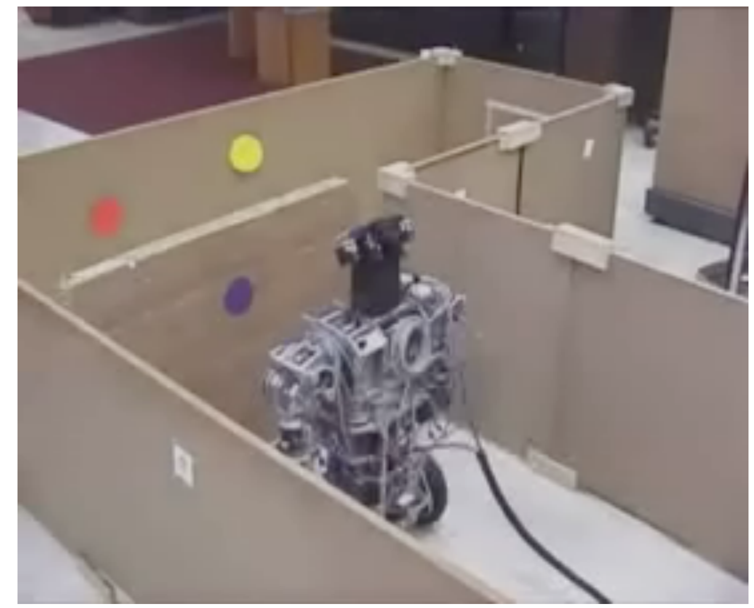
An example with multiple start positions.

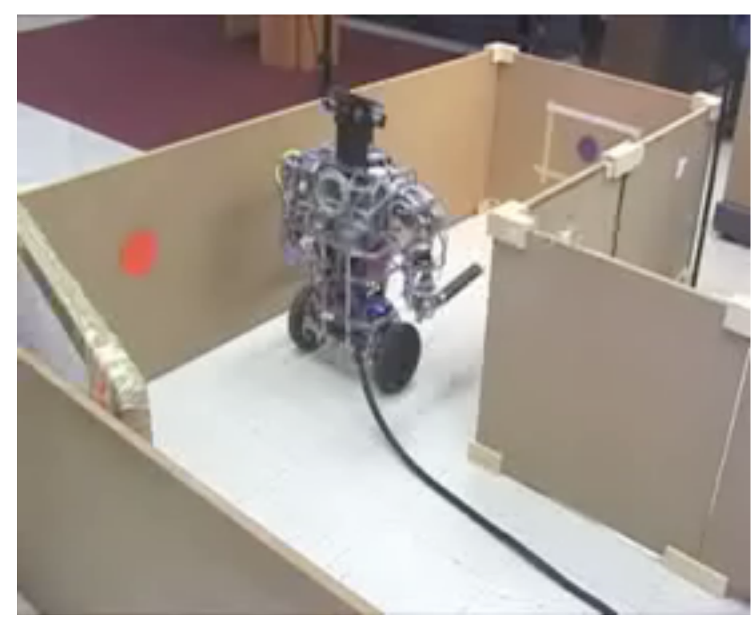


Scaling Up

Combine skill chaining with skill-specific abstractions.



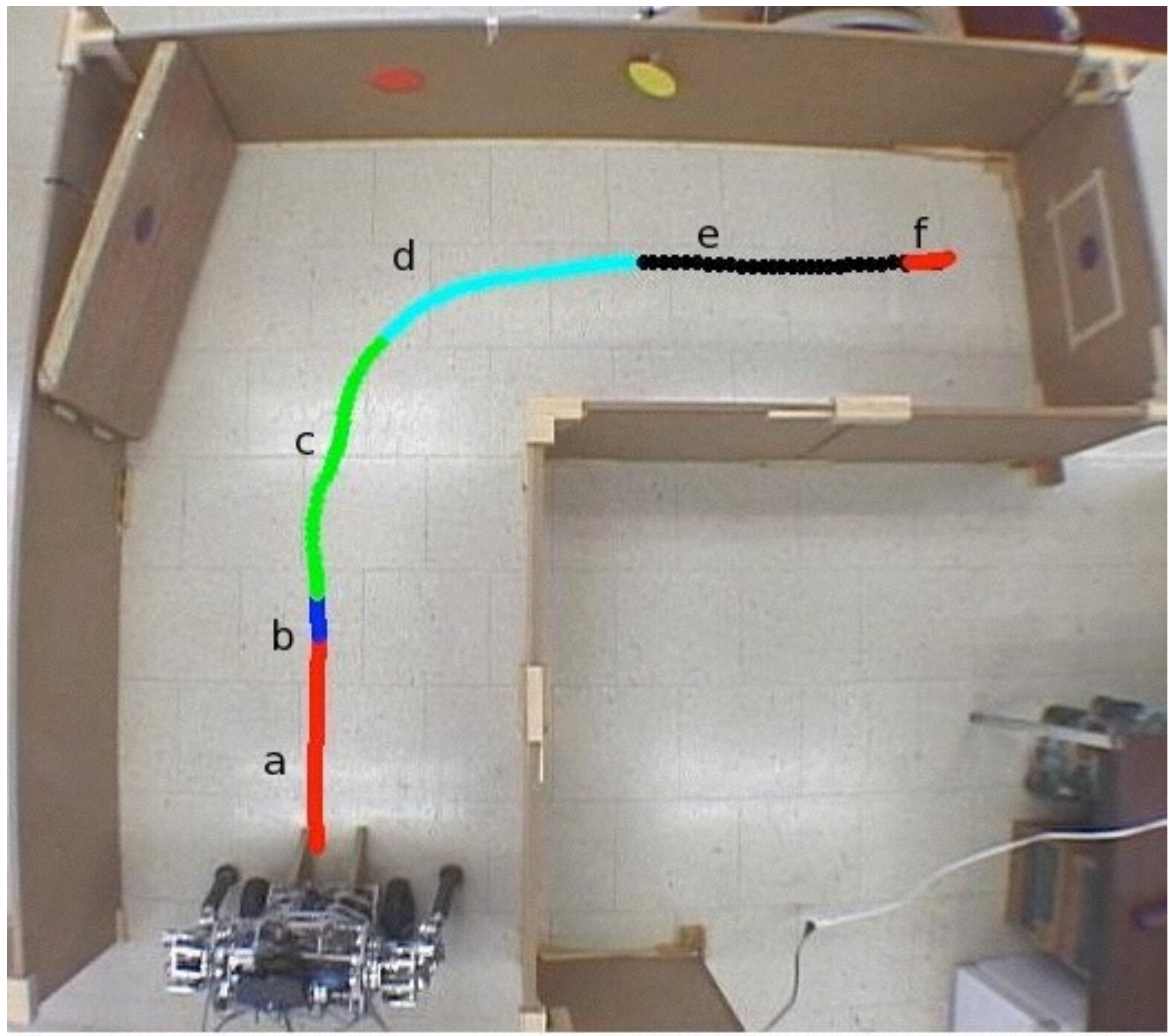




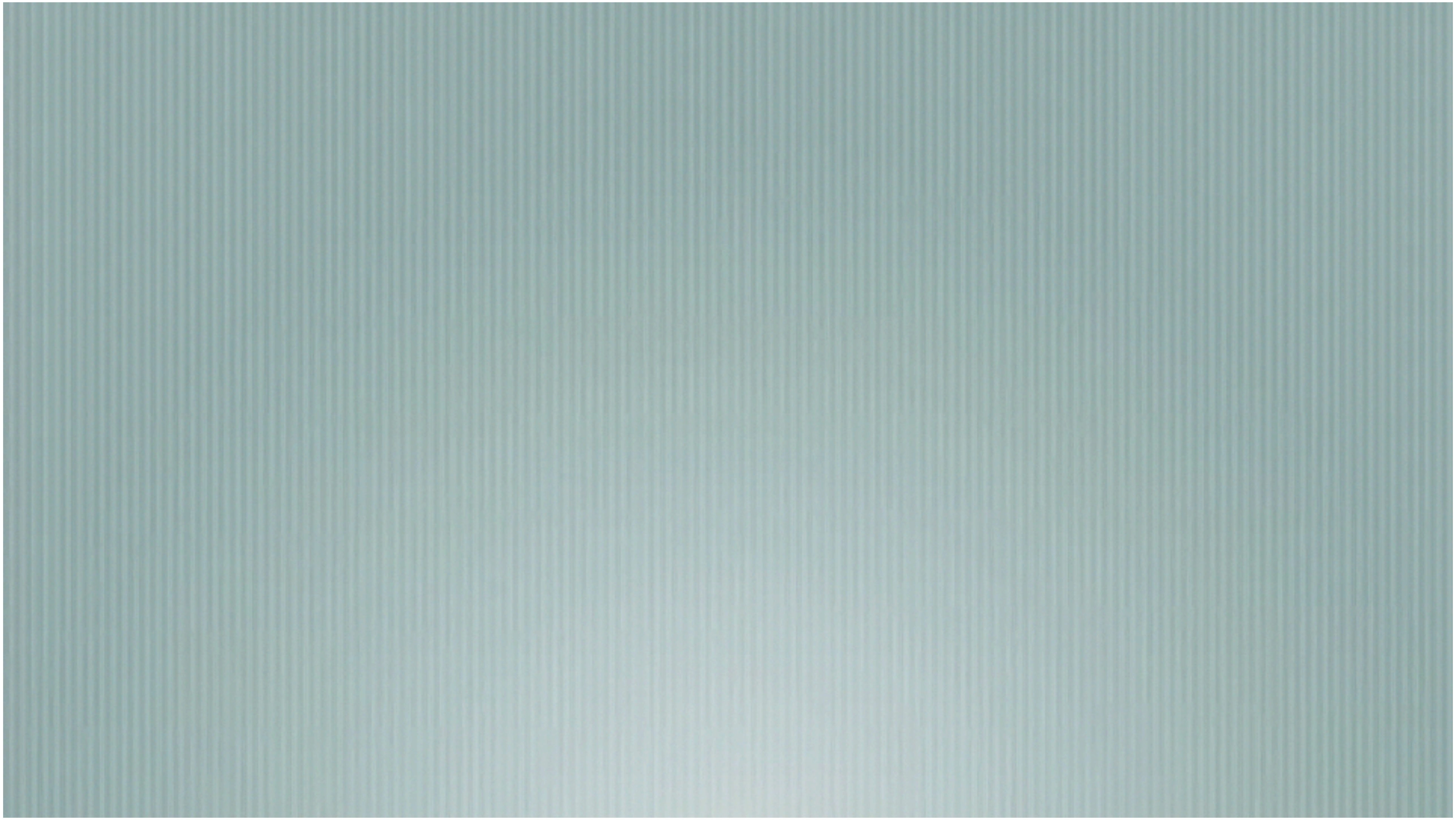
(Konidaris, Kuindersma, Grupen and Barto, NIPS 2010)

CST on the uBot

Trajectory segmented into appropriate skills + abstractions.



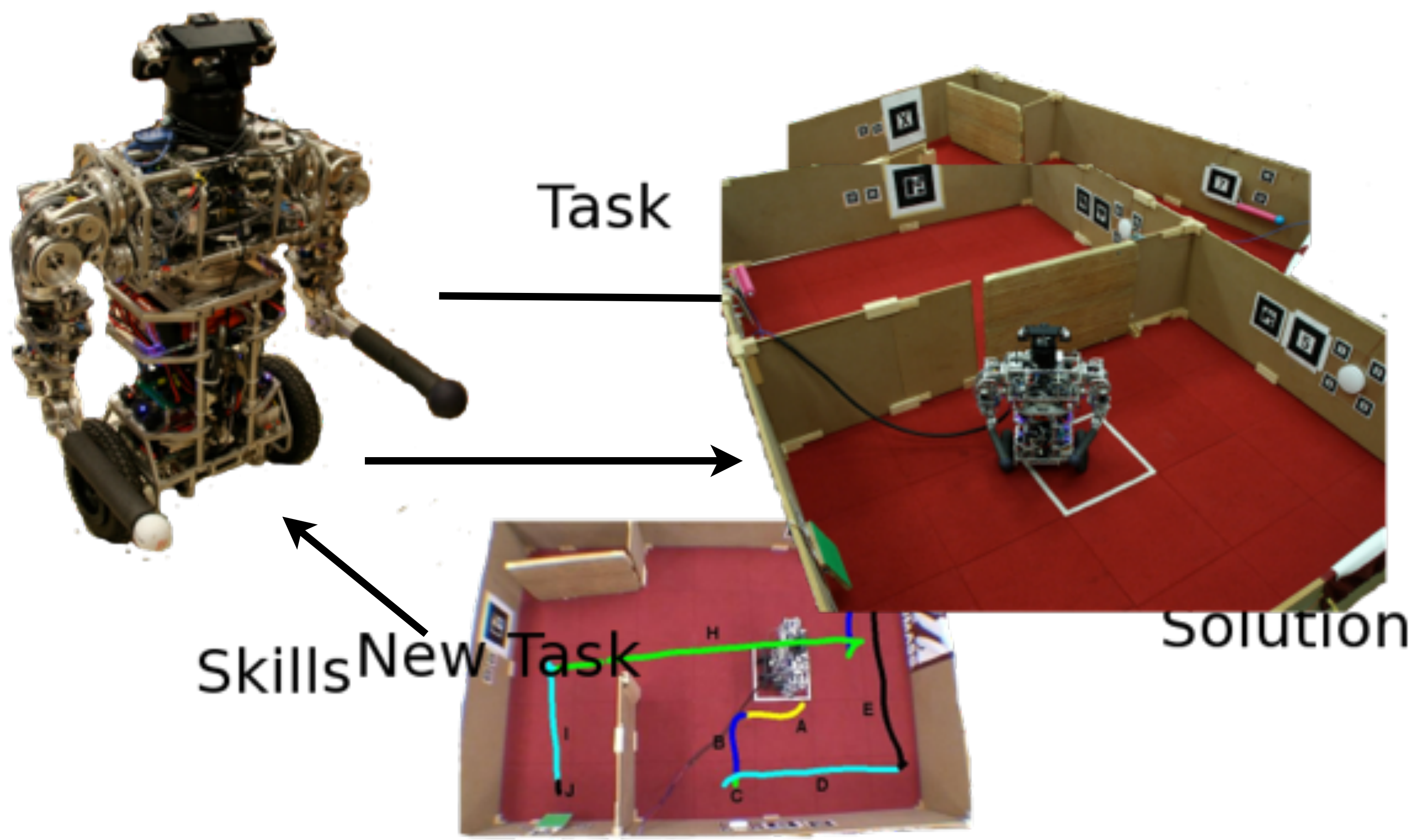
Follow-on Work



ARSA

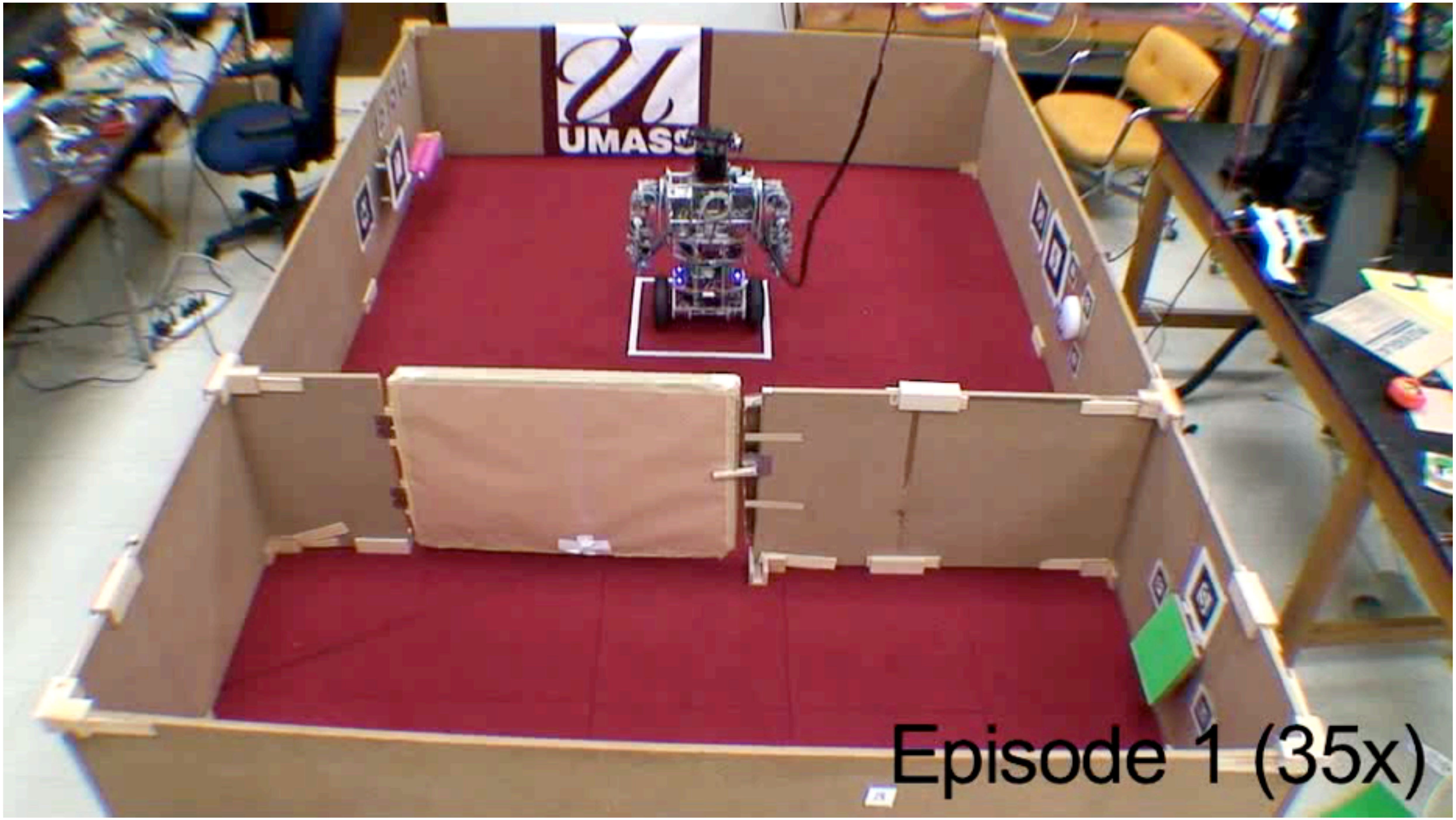
Demonstration of:

- A mobile manipulator learning to solve a task
- Extracting skills from solution
- Deploying them in a new task

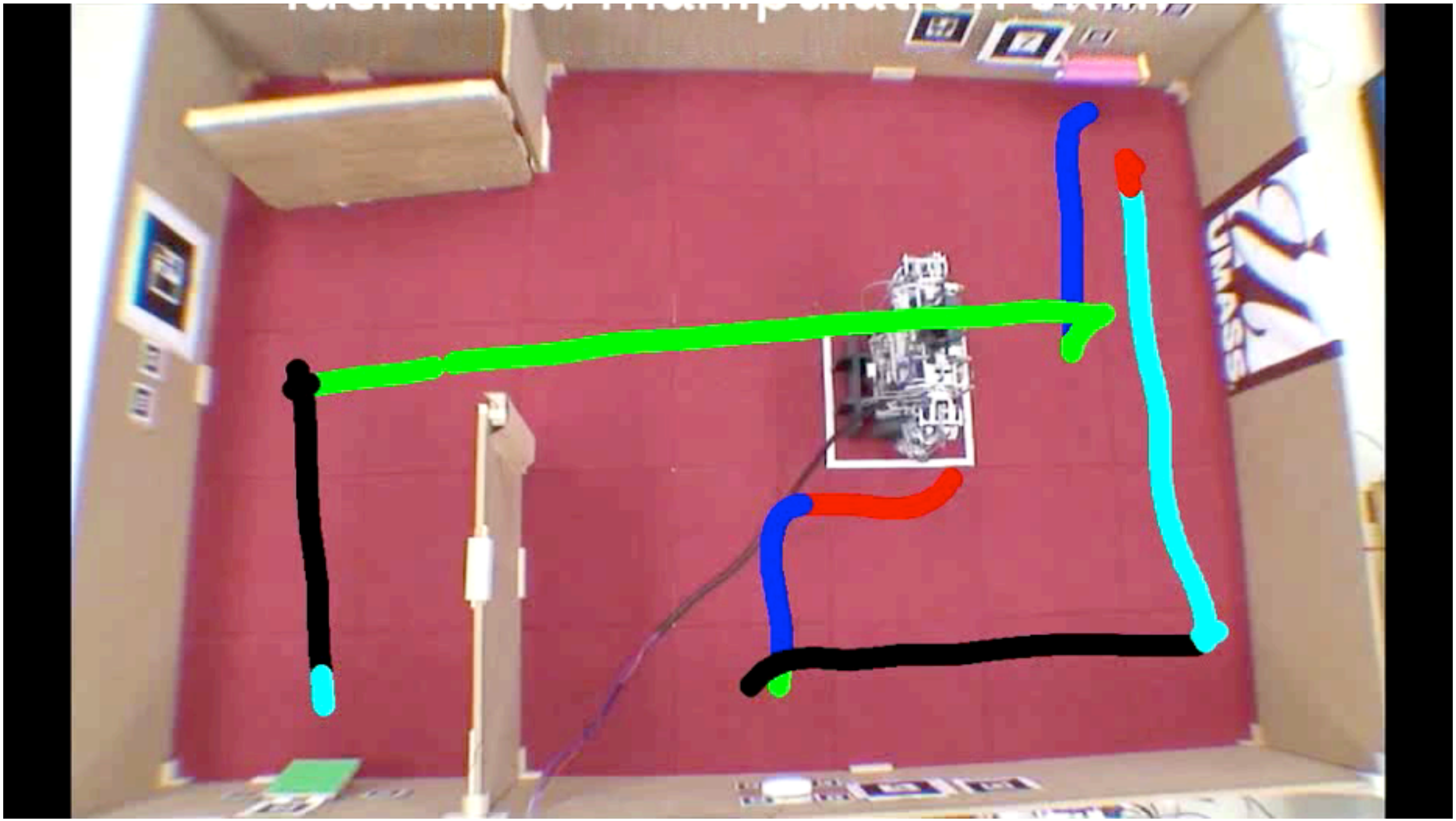


(Konidaris, Kuindersma, Grupen and Barto, AAAI 2011)

Training Room

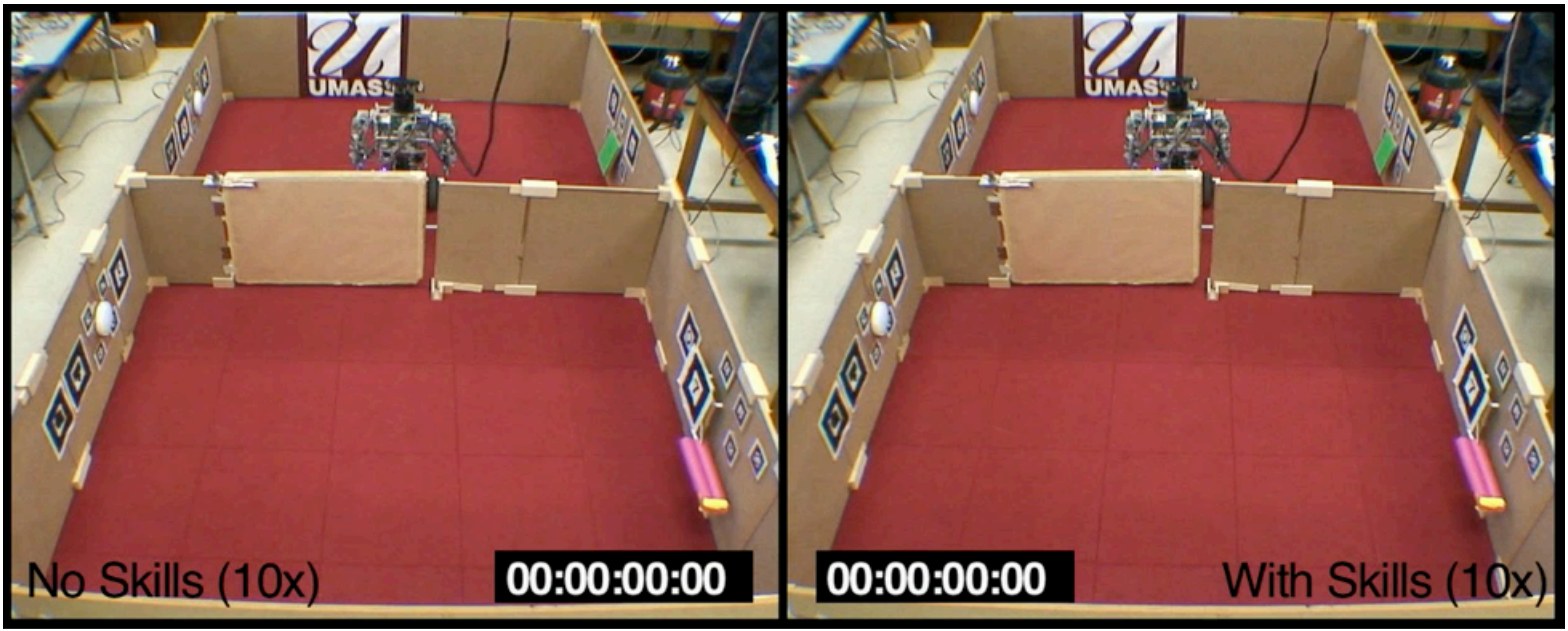


Acquired Skills



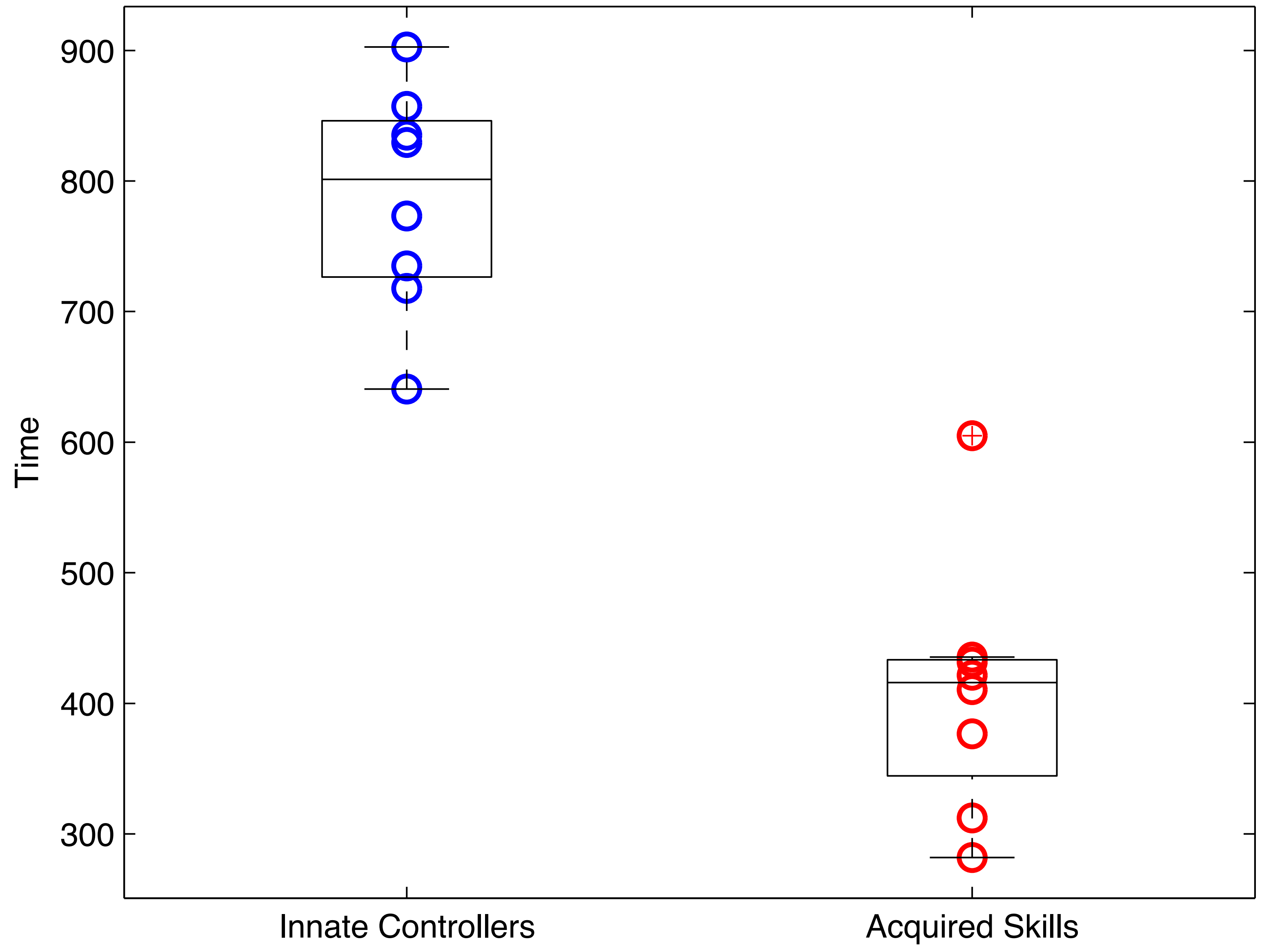
The Test Room

The Test Room



The Test Room

[AAAI 2011]



Summary

Scaled skill acquisition to mobile manipulator:

- Skills extracted because they are useful
- Suitable for further learning (individually)
- Suitable for deployment in new problems

Acquired skills can improve a robot's problem-solving abilities.

Meta-Summary

HRL, and options in particular, provides a framework for:

- Learning and planning with high-level actions.
- Discovering high-level actions from experience.

Key aspects to scaling up:

- Adaptively break complex tasks into simple ones.
- Skill-specific abstractions.
- Skill transfer and reuse.