

# Main ideas of the options work

- Learning/planning with tiny time steps *and* longer temporal scales
- Things postponed, not considered, cleverly avoided
  - ❖ State abstraction
  - ❖ State induction (non-Markov, figuring out what the states are)
  - ❖ Internal structure of temporal abstractions (just two levels)
  - ❖ Unstoppable options, immutable options
- Levels of ideas in this work
  - ❖ The option-to-option (SMDP) level, unstoppable
  - ❖ Intra-option level, look inside option *execution* to more efficiently compute/learn the external view of an option (its value and model)
  - ❖ Look inside options to create new options
    - E.g., interruption
    - Keep the immutable “unstoppable” view
  - ❖ Learn the options themselves  $(\pi, l, \beta)$

# **Between MDPs and Semi-MDPs: A framework for Temporal Abstraction**

**Richard S. Sutton**  
University of Alberta

*with*

Satinder Singh, Doina Precup, B. Ravindran

*and thanks to*

Andy Barto, Amy McGovern, Andrew Fagg  
Ron Parr, Csaba Szepesvari

# The Problem of Temporal Abstraction

How do we connect the high level to the low-level?

" the human level to the physical level?

" the decide level to the action level?

MDPs are great, search is great,

excellent rep'ns of decision-making, choice, outcome

but they are too flat

Can we keep their elegance, clarity, and simplicity,  
while connecting and crossing levels?

# Goal: Extend RL framework to temporally abstract action

- While **minimizing changes** to
  - ❖ Value functions
  - ❖ Bellman equations
  - ❖ Models of the environment
  - ❖ Planning methods
  - ❖ Learning algorithms
- While **maximizing generality**
  - ❖ General dynamics and rewards
  - ❖ Ability to express all courses of behavior
  - ❖ Minimal commitments to other choices
    - Execution, e.g., hierarchy, interruption, intermixing with planning
    - Planning, e.g., incremental, synchronous, trajectory based, “utility” problems
    - State abstraction and function approximation
    - Creation/Constructivism

*It's a  
dimensional  
thing*

# Related Work

## *“Classical” AI*

Fikes, Hart & Nilsson(1972)  
Newell & Simon (1972)  
Sacerdoti (1974, 1977)

## *Macro-Operators*

Korf (1985)  
Minton (1988)  
Iba (1989)  
Kibler & Ruby (1992)

## *Qualitative Reasoning*

Kuipers (1979)  
de Kleer & Brown (1984)  
Dejong (1994)

Laird et al. (1986)  
Drescher (1991)  
Levinson & Fuchs (1994)  
Say & Selahatin (1996)  
Brafman & Moshe (1997)

## *Robotics and Control Engineering*

Brooks (1986)  
Maes (1991)  
Koza & Rice (1992)  
Brockett (1993)  
Grossman et. al (1993)  
Dorigo & Colombetti (1994)  
Asada et. al (1996)  
Uchibe et. al (1996)  
Huber & Grupen(1997)  
Kalmar et. al (1997)  
Mataric(1997)  
Sastry (1997)  
Toth et. al (1997)

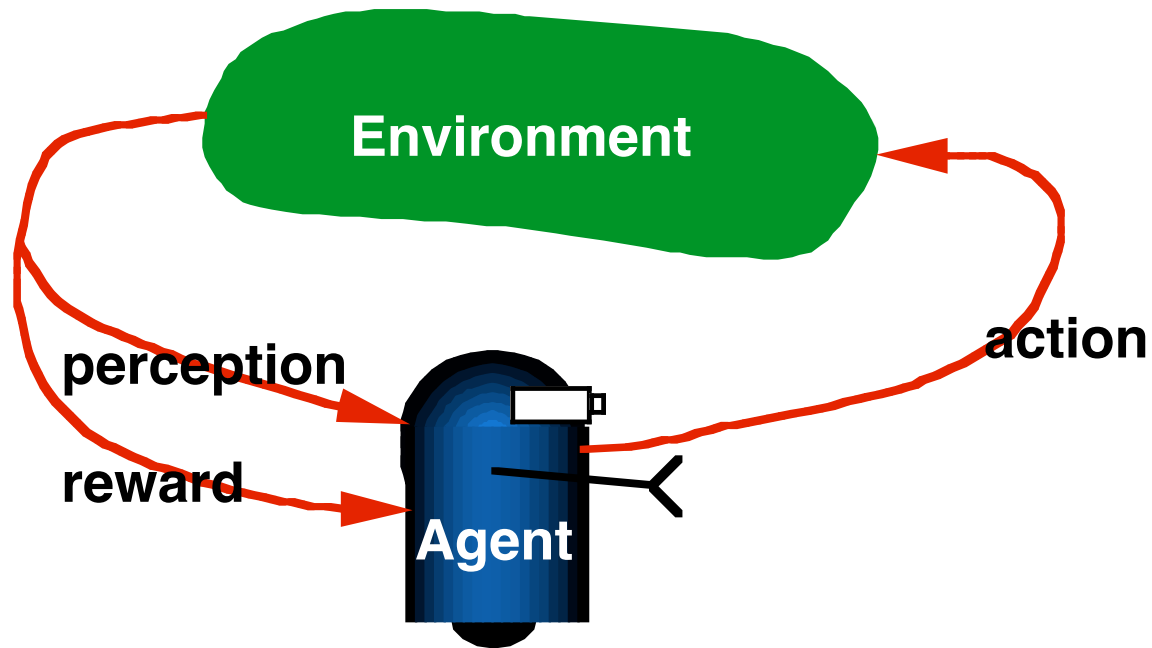
## *Reinforcement Learning and MDP Planning*

Mahadevan & Connell (1992)  
Singh (1992)  
Lin (1993)  
Dayan & Hinton (1993)  
Kaelbling(1993)  
Chrisman (1994)  
Bradtke & Duff (1995)  
Ring (1995)  
Sutton (1995)  
Thrun & Schwartz (1995)  
Boutillier et. al (1997)  
Dietterich(1997)  
Wiering & Schmidhuber (1997)  
Precup, Sutton & Singh (1997)  
McGovern & Sutton (1998)  
Parr & Russell (1998)  
Drummond (1998)  
Hauskrecht et. al (1998)  
Meuleau et. al (1998)  
Ryan and Pendrith (1998)

# Outline

- The RL (MDP) framework
- The extension to temporally abstract “options”
  - ❖ Options and Semi-MDPs
  - ❖ Hierarchical planning and learning
- Rooms example
- Between MDPs and Semi-MDPs
  - ❖ Improvement by interruption (including Spy plane demo)
  - ❖ A taste of
    - Intra-option learning
    - Subgoals for learning options
    - RoboCup soccer demo

# RL is Learning from Interaction



**RL is like Life!**

- complete agent
- temporally situated
- continual learning and planning
- object is to affect environment
- environment is stochastic and uncertain

# More Formally: Markov Decision Problems (MDPs)

An MDP is defined by  $\langle S, A, p, r, \gamma \rangle$

$S$  – set of **states** of the environment

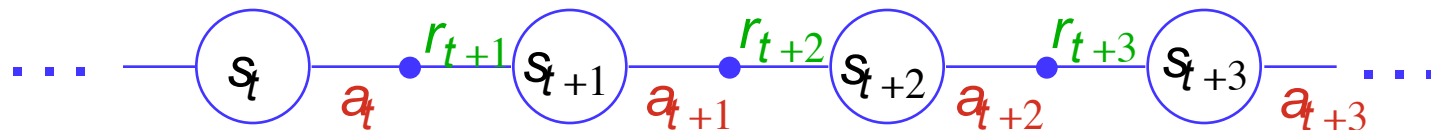
$A_s$  – set of **actions** possible in state  $s$

$p_{ss'}^a$  – **probability of transition** from  $s$  to  $s'$  given action  $a$

$r_s^a$  – **expected reward** when executing action  $a$  in state  $s$

$\gamma$  – **discount rate** for expected reward

time is **discrete**,  $t = 0, 1, 2, \dots$



# The Objective – Maximize Value

Find a **policy**  $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$  (a way of acting) that maximizes *expected total discounted reward* from every state (value)

$$V^\pi(\mathbf{s}) = \mathbb{E} \left\{ r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \mid \mathbf{s}_0 = \mathbf{s}, \pi \right\}$$

achieving

$$V^*(\mathbf{s}) = \max_{\pi} V^\pi(\mathbf{s})$$

**Value functions**

btw

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E} \left\{ r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}, \pi \right\}$$

# Options – Temporally Abstract Actions

An option is a triple,  $o = \langle I, \pi, \beta \rangle$

$I \subseteq S$  is the subset of states in which  $o$  may be initiated

$\pi : S \times A \rightarrow [0,1]$  is the policy followed during  $o$

$\beta : S \rightarrow [0,1]$  is the probability of terminating in each state

Execution is nominally **hierarchical** (call-and-return)

E.g., the **docking** option:

$I$  : all states in which charger is in sight

$\pi$  : hand-crafted controller

$\beta$  : terminate when docked or charger not visible

...there are also “semi-Markov” options

# Options are like actions

Just as a state has a set of actions,  $A_s$ , with  $A = \cup A_s$

It also has a set of options,  $O_s$ , with  $O = \cup O_s$

Just as we can have a **flat policy**, over actions,  $\pi : S \times A \rightarrow [0,1]$

We can have a **hierarchical policy**, over *options*,  $\mu : S \times O \rightarrow [0,1]$

To execute  $\mu$  in  $s$  :

- select option  $o$  with probability  $\mu(s,o)$

- follow  $o$  until it terminates, in  $s'$

- then choose from  $\mu(s',o)$  again, and so on until termination

Every hierarchical policy determines a flat policy

$$\pi = f(\mu)$$

Even if all the options  $o \in O$  are Markov,  $f(\mu)$  is usually not Markov

Actions are a special case of options

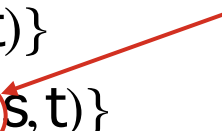
# Value Functions for Options (hierarchical value functions)

Define value functions for options, similar to the MDP case

$$V^\mu(s) = E \{r_{t+1} + \gamma r_{t+2} + \dots \mid E(\mu, s, t)\}$$

$$Q^\mu(s, o) = E \{r_{t+1} + \gamma r_{t+2} + \dots \mid E(o, \mu, s, t)\}$$

sequential  
composition



Now consider policies  $\mu \in \Pi(O)$  restricted to choose only from options in  $O$  :

$$V_O^*(s) = \max_{\mu \in \Pi(O)} V^\mu(s)$$

$$Q_O^*(s, o) = \max_{\mu \in \Pi(O)} Q^\mu(s, o)$$

# “Option Models” of the Environment

Planning requires models of the consequences of action

The model of taking an option  $o$  from state  $s$  has

- a reward part

$$r_s^o = E \left\{ r_1 + \gamma r_2 + \gamma^2 r_3 + \dots + \gamma^{k-1} r_k \mid s_0 = s, o \text{ initiated in } s_0 \right\}$$

- and a next state part

$k$  = time at which  $o$  ends

$$p_{ss'}^o = \sum_{k=1}^{\infty} \underbrace{p(s', k)} \gamma^k \quad \forall s'$$

Probability of terminating in  $S'$  in  $k$  steps

Just like primitive actions

# Hierarchical (General) Bellman Equations

For **optimal** value functions:

$$V_0^*(s) = \max_{o \in \mathcal{O}} r_s^o + \sum_{s'} p_{ss'}^o V_0^*(s')$$

$$Q_0^*(s, o) = r_s^o + \sum_{s'} p_{ss'}^o \max_{o' \in \mathcal{O}} Q_0^*(s', o')$$

For **policy-specific** value functions:

$$V^\mu(s) = \sum_o \mu(s, o) \left[ r_s^o + \sum_{s'} p_{ss'}^o V^\mu(s') \right]$$

$$Q^\mu(s, o) = r_s^o + \sum_{s'} p_{ss'}^o \sum_{o'} \mu(s', o') Q^\mu(s', o')$$

# Hierarchical (General) Value Iteration

$$\text{Initialize: } V_0(s) \leftarrow 0 \quad \forall s \in S$$

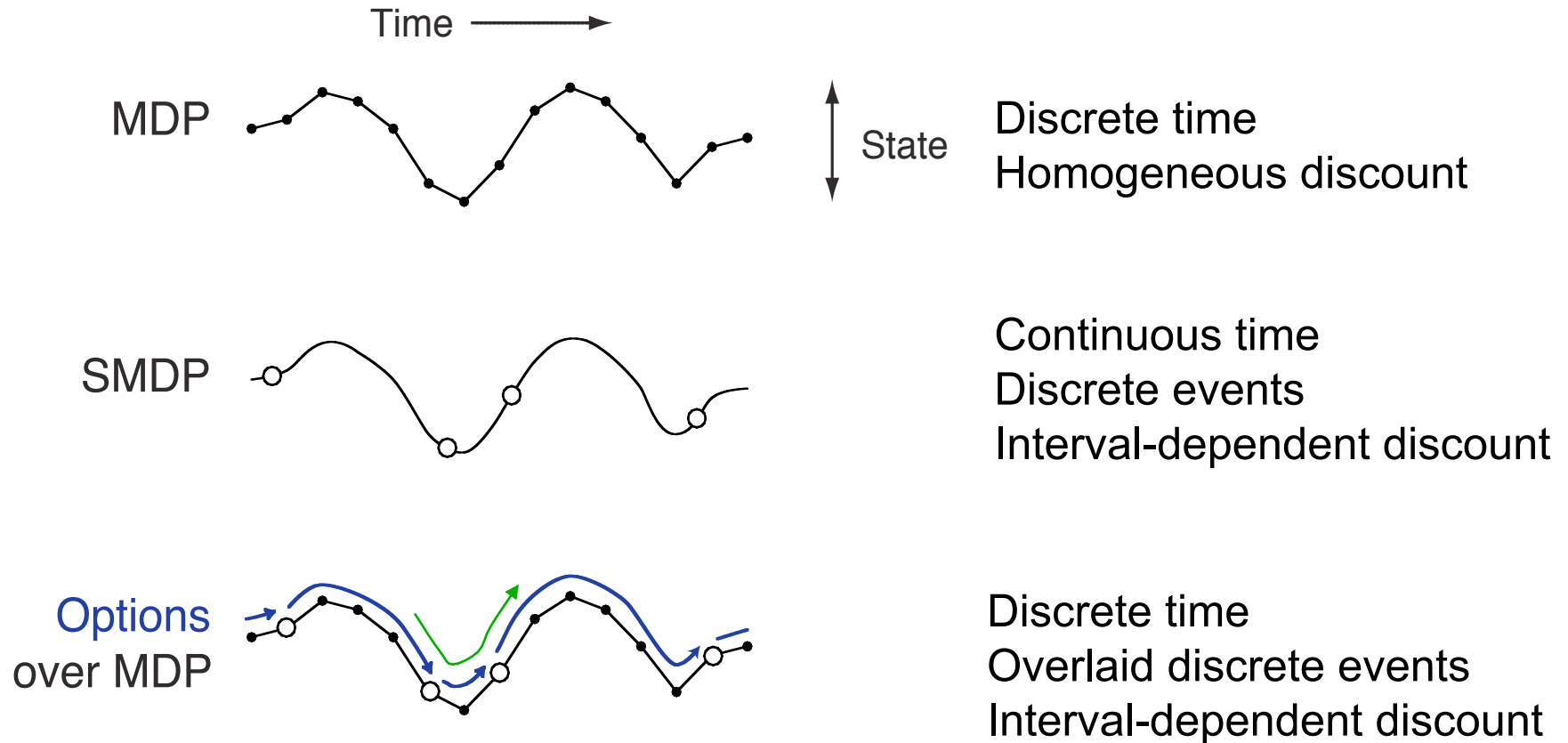
$$\text{Iterate: } V_{k+1}(s) \leftarrow \max_{o \in O} \left[ r_s^o + \sum_{s' \in S} p_{ss'}^o V_k(s') \right] \quad \forall s \in S$$

$$\lim_{k \rightarrow \infty} V_k = V_0^*$$

$$\mu_0^*(s) = \arg \max_{o \in O} r_s^o + \sum_{s'} p_{ss'}^o V_0^*(s') \quad (\text{general optimal policy})$$

Reduces to conventional value iteration if  $A = O$

# Options define a Semi-Markov Decision Process (SMDP)



A discrete-time SMDP overlaid on an MDP.  
Can be analyzed at either level.

# MDP + Options = SMDP

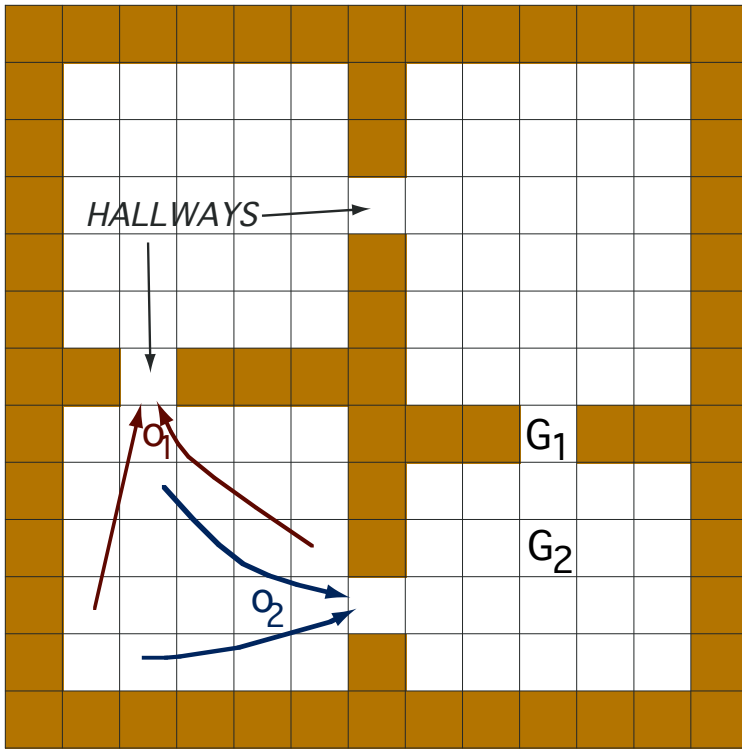
*Theorem:*

Parr, 1998

*For any MDP,  
and any set of options,  
the decision process that chooses among the options,  
executing each to termination,  
is an SMDP.*

Thus all Bellman equations and DP results extend for value functions over options and models of options.  
(we inherit all SMDP theory)

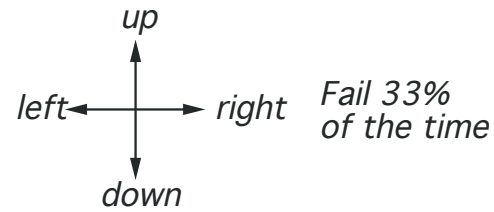
# Rooms Example



All rewards zero,  
except +1 into goal

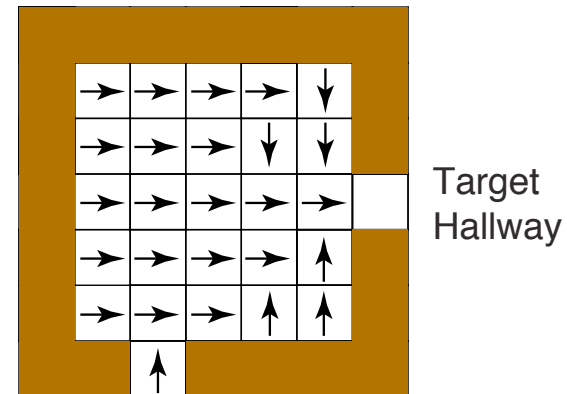
$\gamma = .9$

4 stochastic  
primitive actions



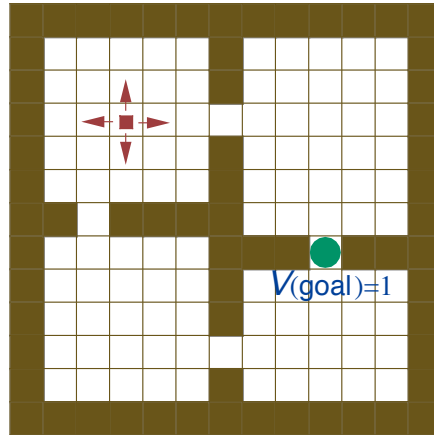
8 multi-step options  
(to each room's 2 hallways)

Policy of  
one option:

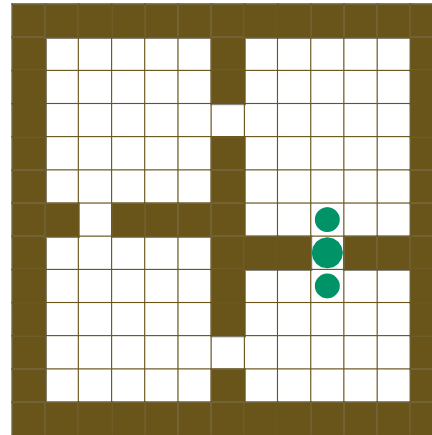


# Planning (SVI) in Rooms Example

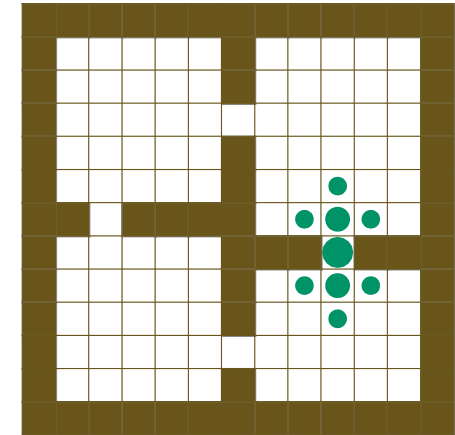
with cell-to-cell primitive actions



Iteration #0

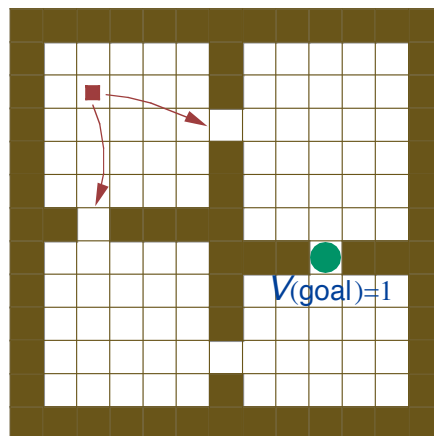


Iteration #1

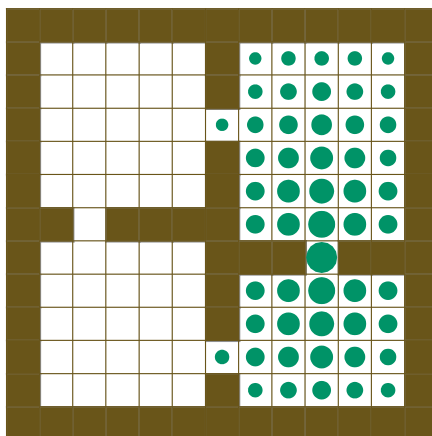


Iteration #2

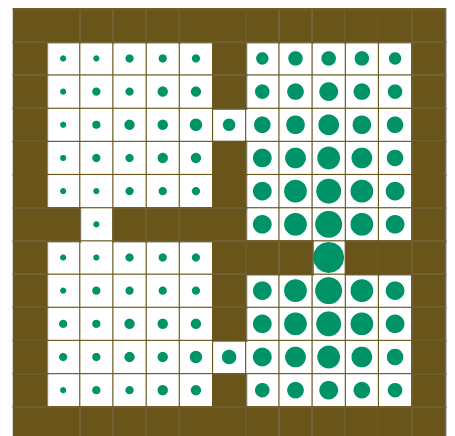
with room-to-room options



Iteration #0

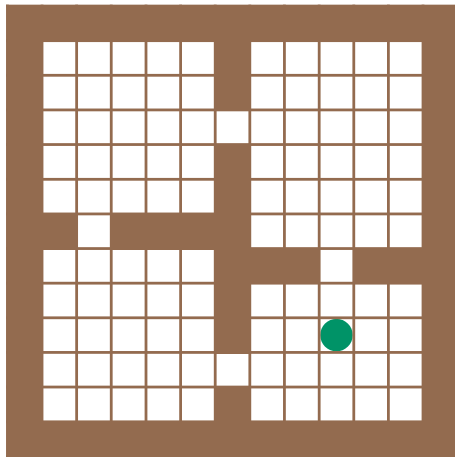


Iteration #1

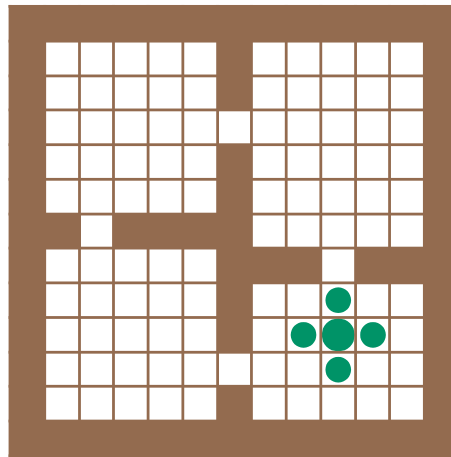


Iteration #2

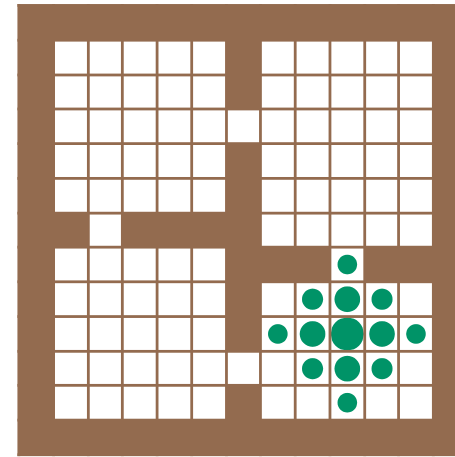
# Planning with Goal $\neq$ Subgoal both primitive actions and options



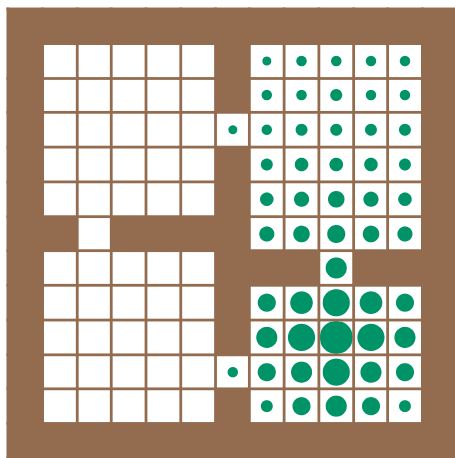
Initial values



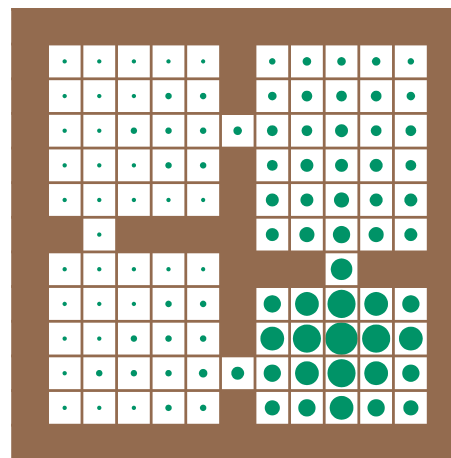
Iteration #1



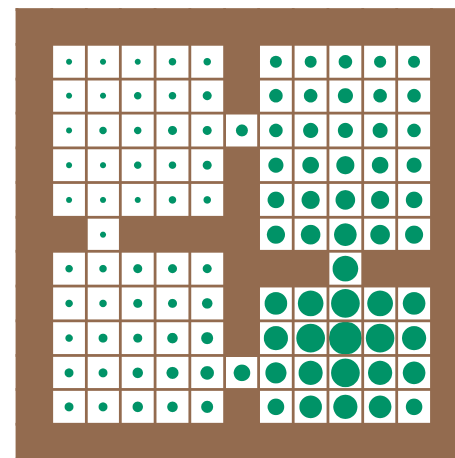
Iteration #2



Iteration #3

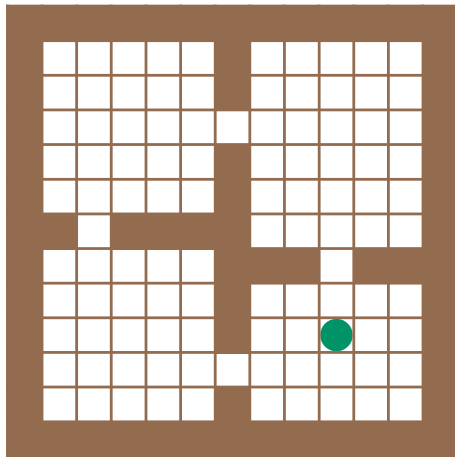


Iteration #4

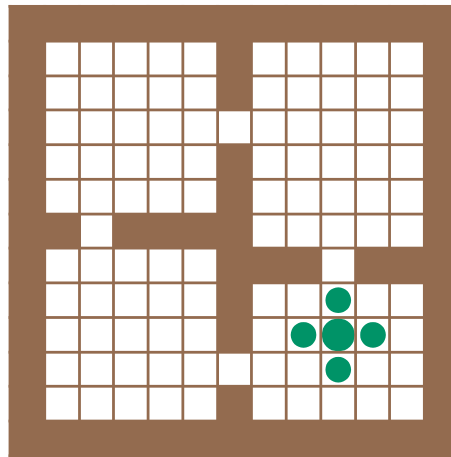


Iteration #5

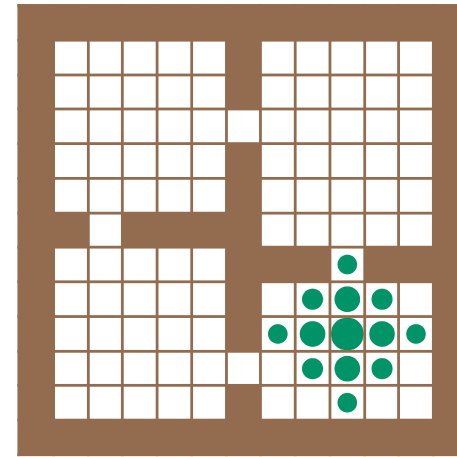
# Planning with Goal $\neq$ Subgoal both primitive actions and options



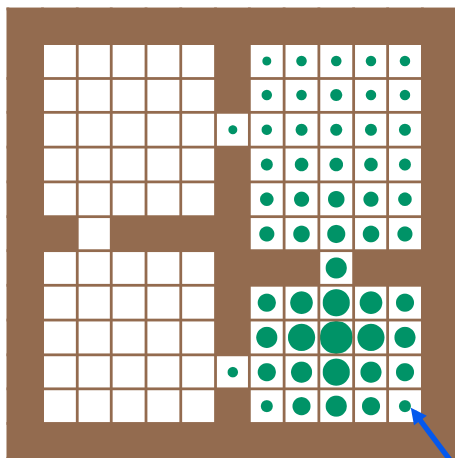
Initial values



Iteration #1

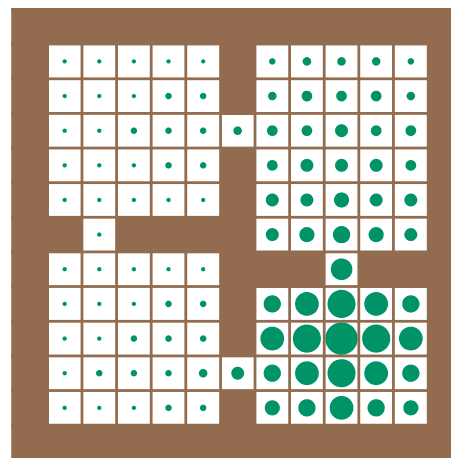


Iteration #2

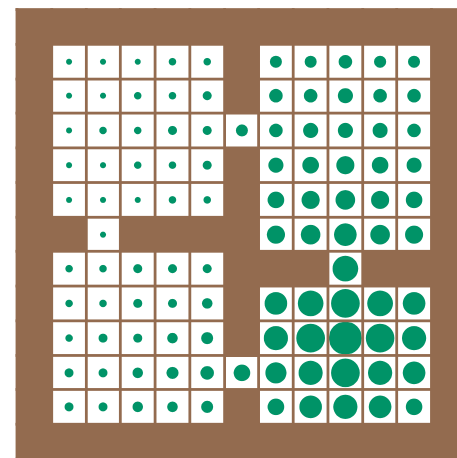


Iteration #3

why?

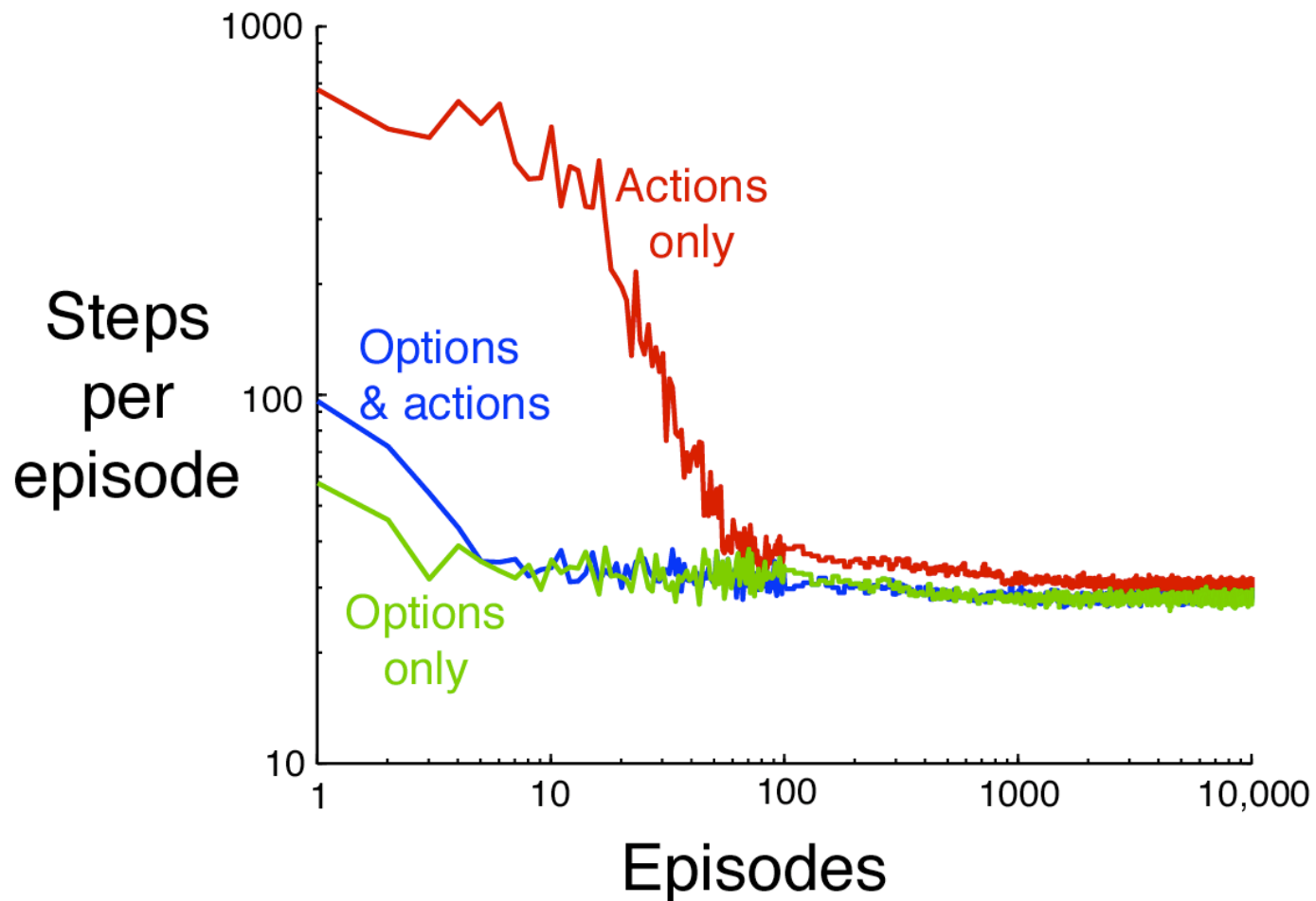


Iteration #4



Iteration #5

# *Learning Path-to-Goal with and without Halfway Options*



# SMDP Theory Provides a lot

- Policies over options:  $\mu : S \times O \mapsto [0,1]$
- Value functions over options:  $V^\mu(s), Q^\mu(s,o), V_0^*(s), Q_0^*(s,o)$
- Learning methods: Bradtke & Duff (1995), Parr (1998)
- Models of options
- Planning methods: e.g. value iteration, policy iteration, Dyna...
- A coherent theory of learning and planning with courses of action at variable time scales, yet at the same level

But the most interesting issues are beyond SMDPs...

# Outline

- The RL (MDP) framework
- The extension to temporally abstract “options”
  - ❖ Options and Semi-MDPs
  - ❖ Hierarchical planning and learning
- Rooms example
- • Between MDPs and Semi-MDPs
  - ❖ Improvement by interruption (including Spy plane demo)
  - ❖ A taste of
    - Intra-option learning
    - Subgoals for learning options
    - RoboCup soccer demo

# Interruption

Idea: We can do better by *sometimes interrupting ongoing options*  
- forcing them to terminate before  $\beta$  says to

Theorem: For any hierarchical policy  $\mu : S \times O \mapsto [0,1]$ ,  
suppose we interrupt its options one or more times, when

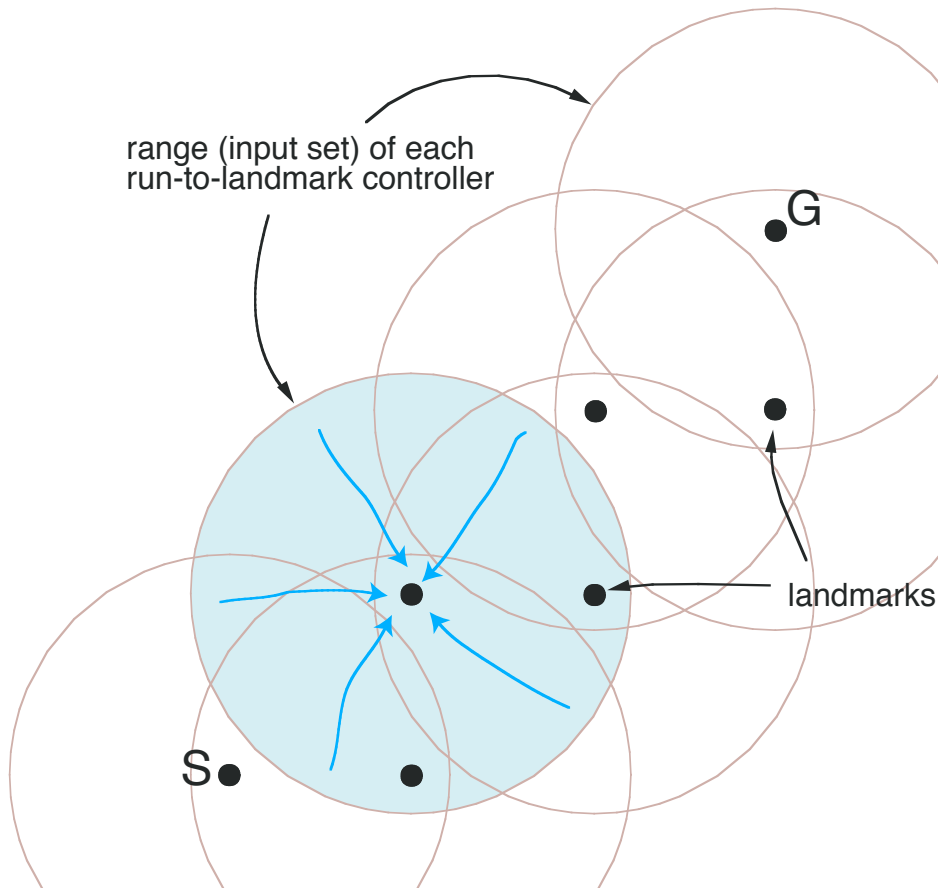
$Q^\mu(s,o) < Q^\mu(s,\mu(s))$ , where  $s$  is the state at that time  
o is the ongoing option  
to obtain  $\mu' : S \times O' \mapsto [0,1]$ ,

Then  $\mu' > \mu$  (it attains more or equal reward everywhere)

Application: Suppose we have determined  $Q_0^*$  and thus  $\mu = \mu_0^*$ .

Then  $\mu'$  is guaranteed better than  $\mu_0^*$   
and is available with no additional computation.

# Landmarks Task



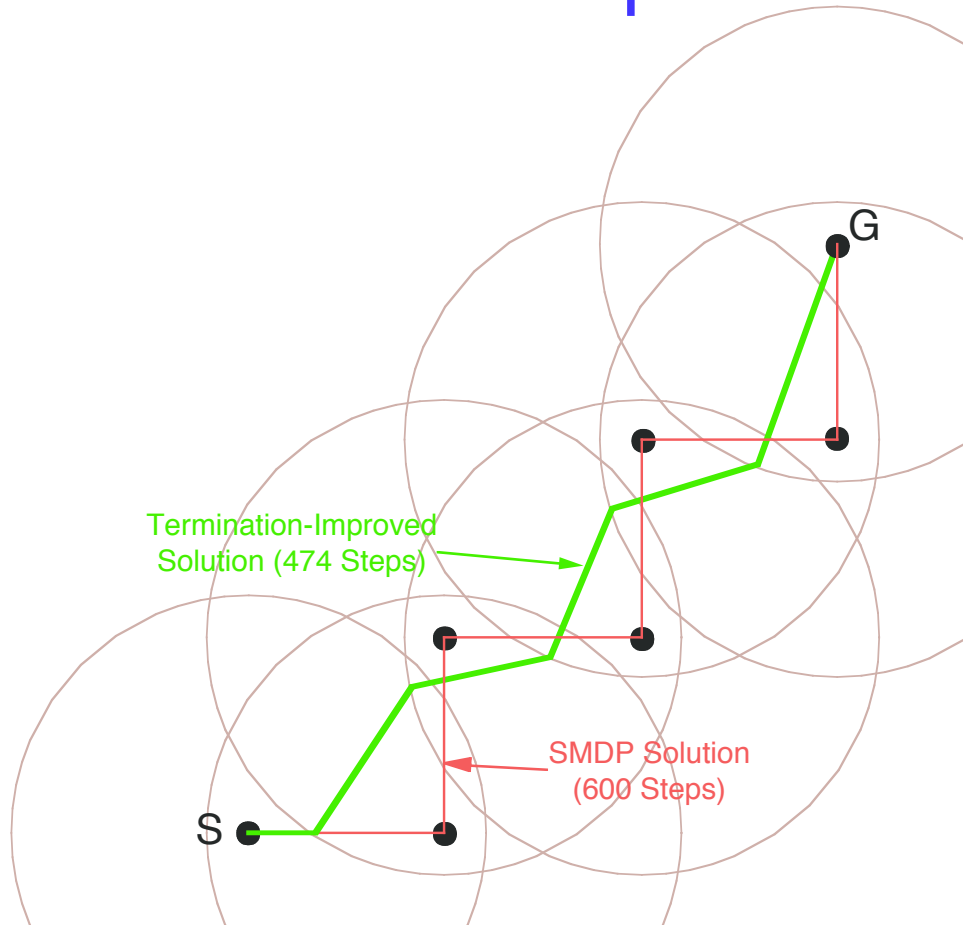
Task: navigate from S to G as fast as possible

4 primitive actions, for taking tiny steps in any direction

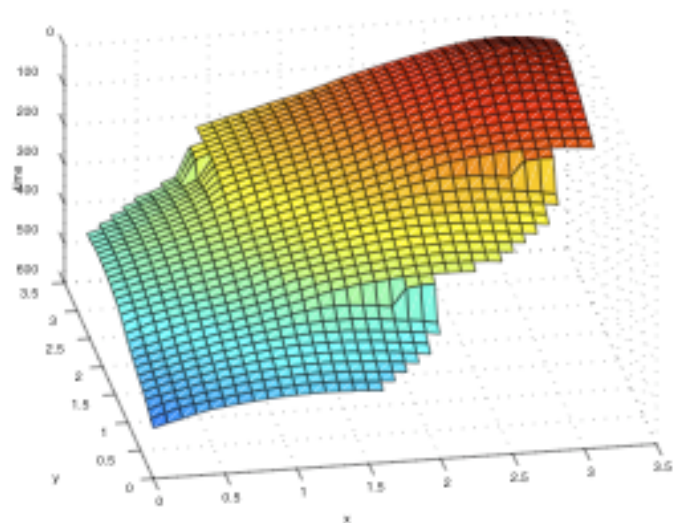
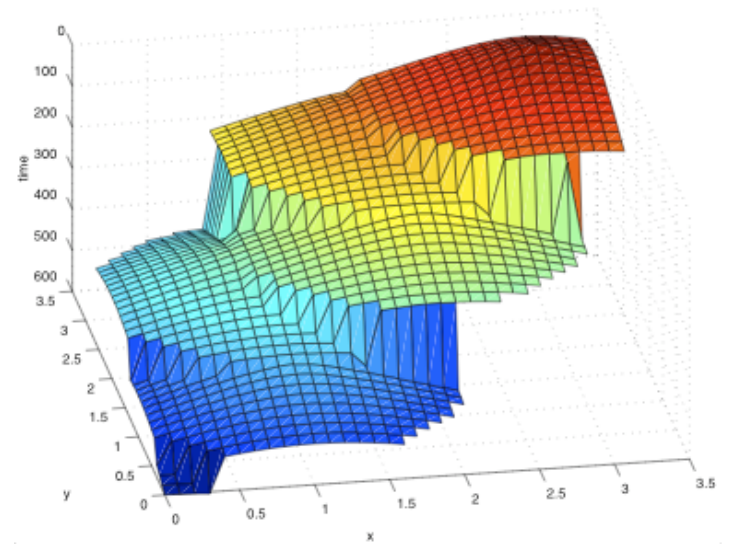
7 controllers for going straight to each one of the landmarks, from within a circular region where the landmark is visible

In this task, planning at the level of primitive actions is computationally intractable, we need the controllers

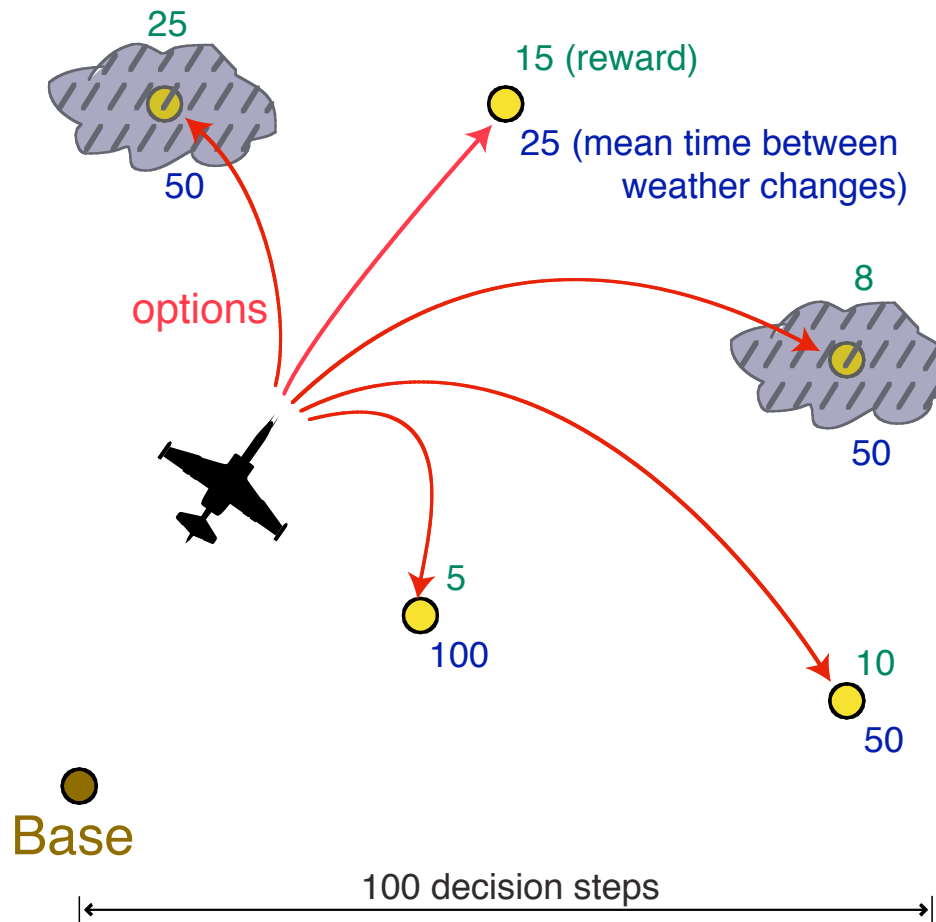
# Termination Improvement for Landmarks Task



Allowing early termination based on models improves the value function at no additional cost!



# Spy Plane Example

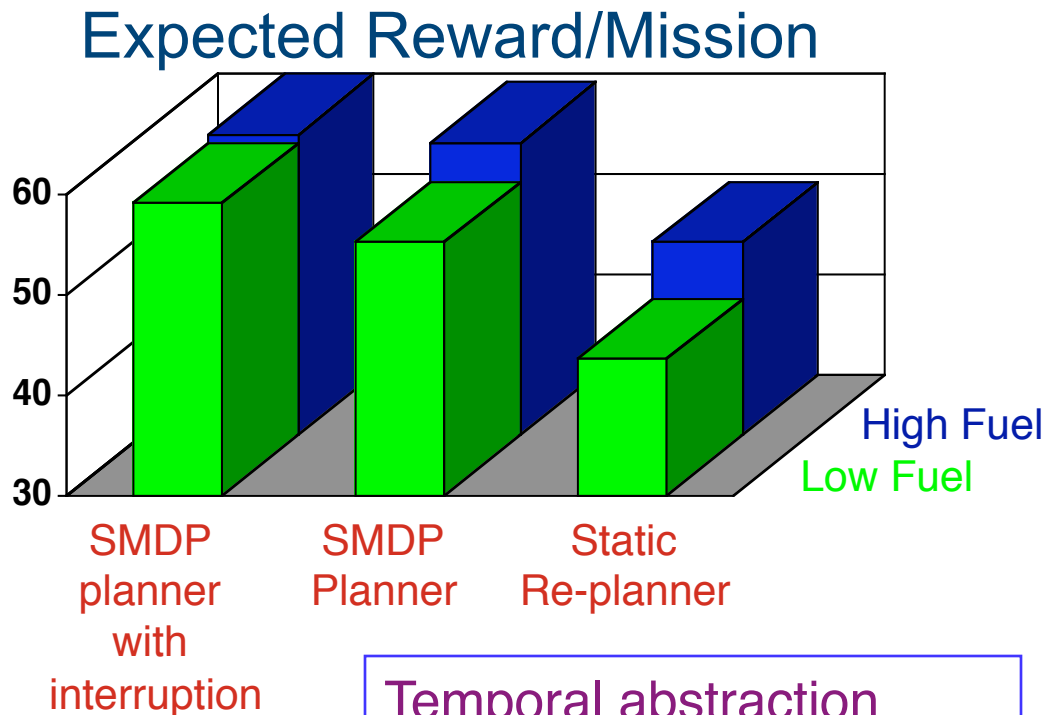


- Mission: Fly over (observe) most valuable sites and return to base
- Stochastic **weather** affects observability (cloudy or clear) of sites
- Limited fuel
- **Intractable** with classical optimal control methods
- Temporal scales:
  - ❖ Actions: which **direction** to fly now
  - ❖ Options: which **site** to head for
- Options compress space and time
  - ❖ Reduce steps from ~600 to ~6

$$Q_0^*(s, o) = r_s^o + \sum_{s'} p_{ss'}^o V_0^*(s')$$

any state  $\sim 10^{10}$       sites only  $\sim 10^6$

# Spy Plane Example (Results)



Temporal abstraction finds better approximation than static planner, with little more computation than SMDP planner

- **SMDP planner:**
  - ❖ Assumes options followed to completion
  - ❖ Plans optimal SMDP solution
- **SMDP planner with interruption**
  - ❖ Plans as if options must be followed to completion
  - ❖ But actually takes them for only one step
  - ❖ Re-picks a new option on every step
- **Static planner:**
  - ❖ Assumes weather will not change
  - ❖ Plans optimal tour among clear sites
  - ❖ Re-plans whenever weather changes

# **UAV Demo**

From McGovern et al., 2000

# Outline

- The RL (MDP) framework
- The extension to temporally abstract “options”
  - ❖ Options and Semi-MDPs
  - ❖ Hierarchical planning and learning
- Rooms example
- Between MDPs and Semi-MDPs
  - ❖ Improvement by interruption (including Spy plane demo)
  - ❖ A taste of
    - Intra-option learning
    - Subgoals for learning options
    - RoboCup soccer demo



# Intra-Option Learning Methods for Markov Options

Idea: take advantage of each **fragment** of experience

## SMDP Q-learning:

- execute option to termination, keeping track of reward along the way
- at the end, **update only the option taken**, based on reward and value of state in which option terminates

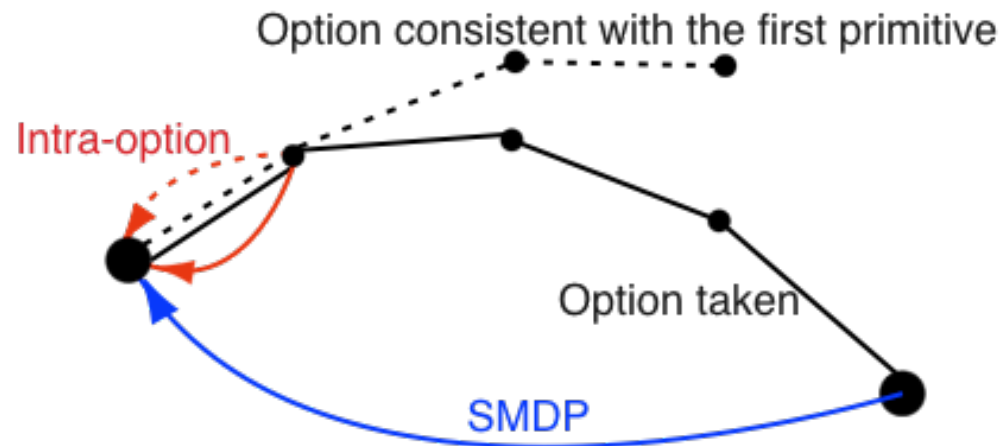
## Intra-option Q-learning:

- after each primitive action, *update all the options that could have taken that action*, based on the reward and the expected value from the next state on

Proven to **converge to correct values**, under same assumptions as 1-step Q-learning

# Intra-Option Learning Methods for Markov Options

Idea: take advantage of each **fragment** of experience

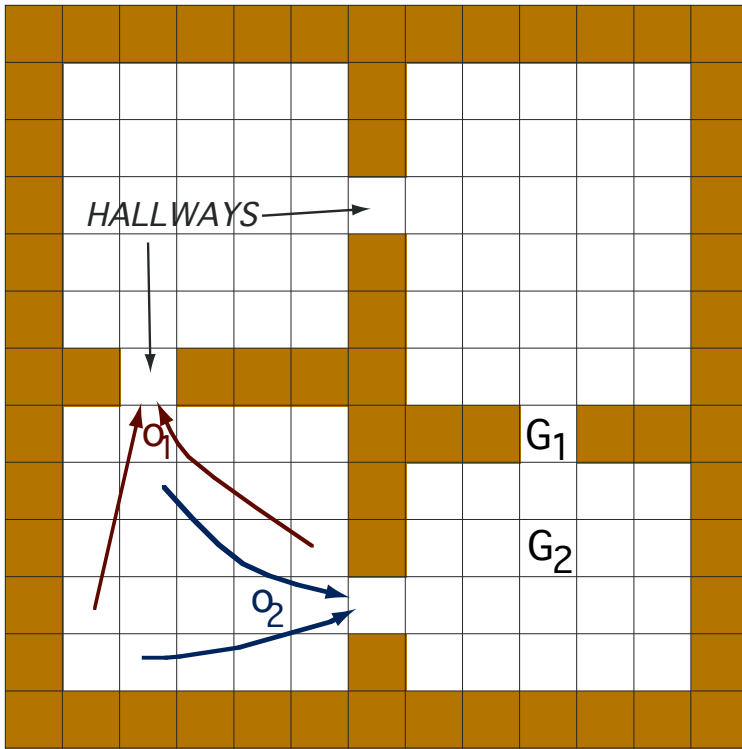


**SMDP Learning:** execute option to termination, then update only the option taken

**Intra-Option Learning:** after each primitive action, update all the options that could have taken that action

Proven to **converge to correct values**, under same assumptions as 1-step Q-learning

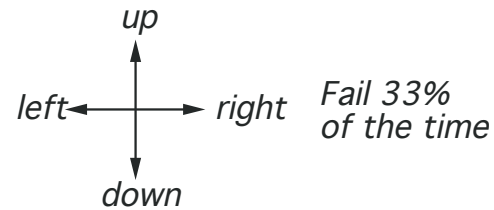
# Returning to the rooms example...



All rewards zero,  
except +1 into goal

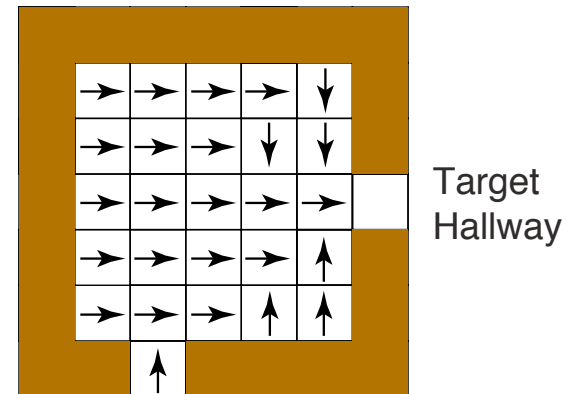
$\gamma = .9$

4 stochastic  
primitive actions

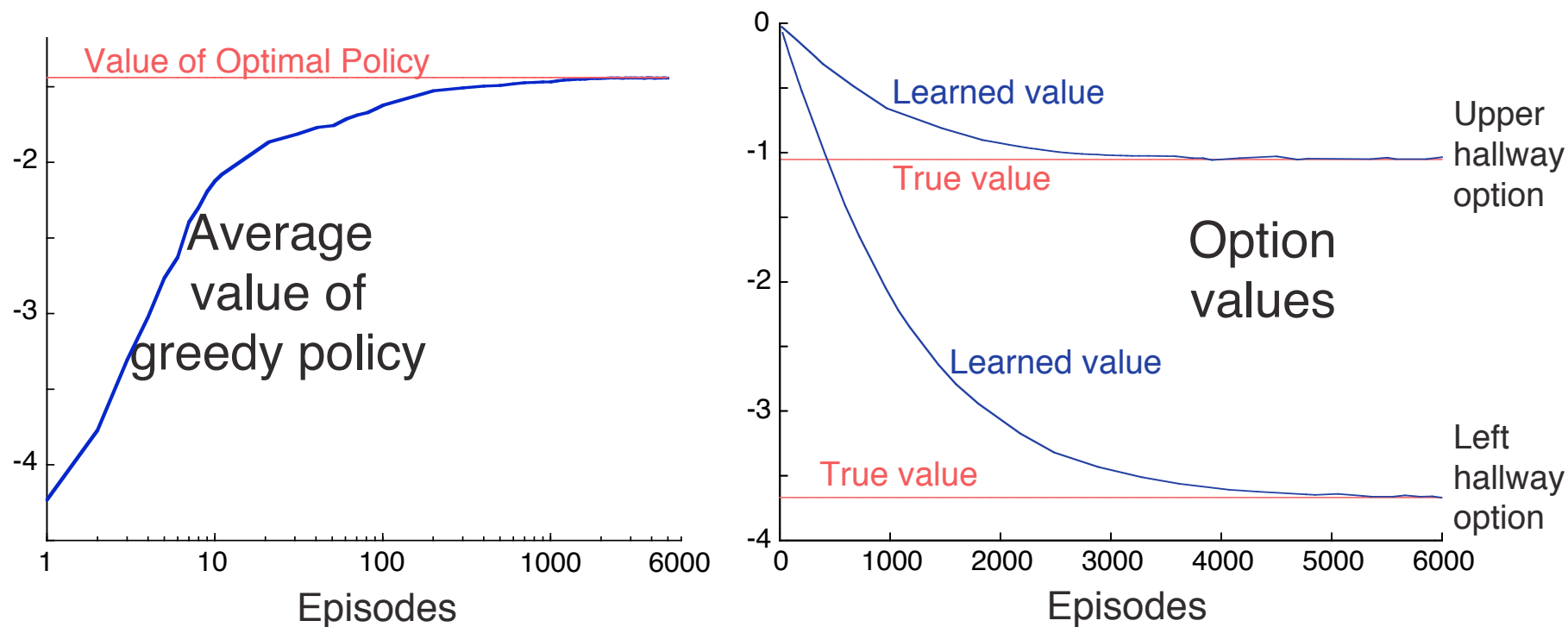


8 multi-step options  
(to each room's 2 hallways)

Policy of  
one option:



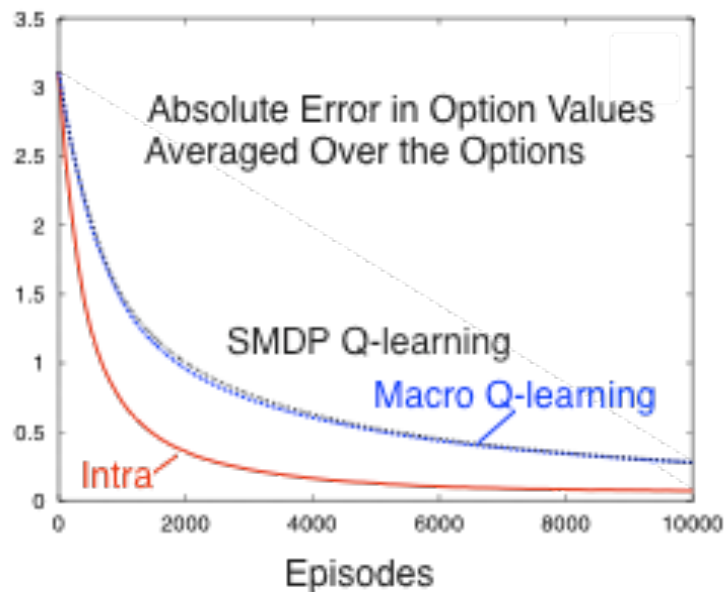
# Intra-Option Value Learning in the Rooms Example



Random start, goal in right hallway, random actions

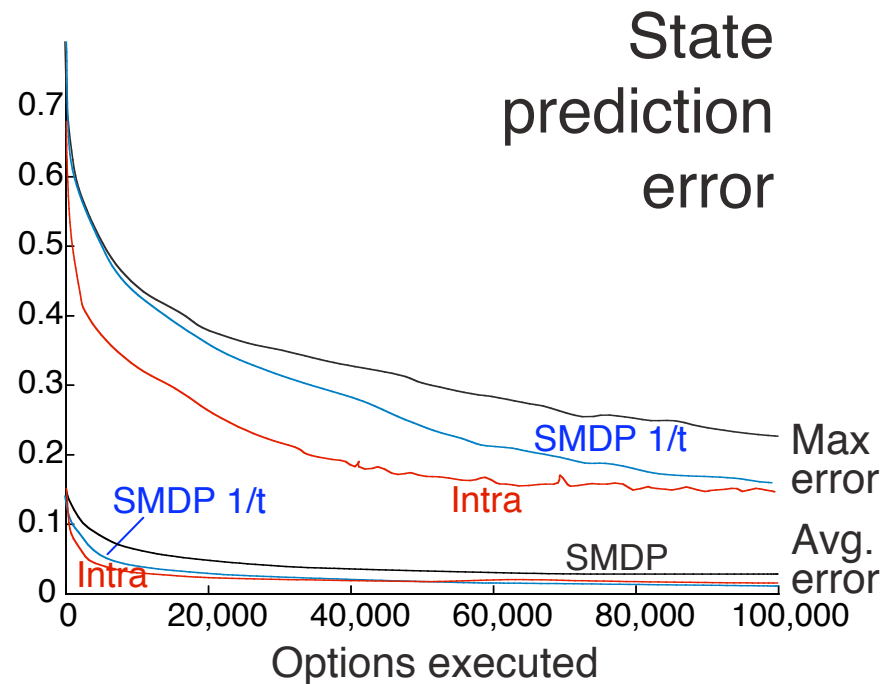
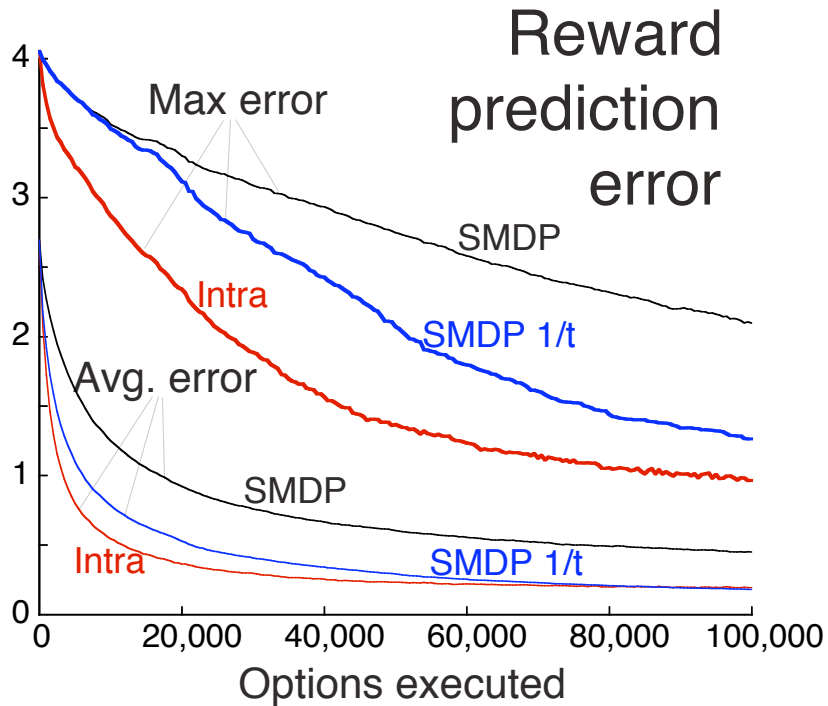
Intra-option methods learn correct values without ever taking the options! SMDP methods are not applicable here

# Intra-Option Value Learning Is Faster Than SMDP Value Learning



Random start, goal in right hallway, choice from A U H, 90% greedy

# Intra-Option Model Learning



Random start state, no goal, pick randomly among all options

**Intra-option methods work much faster than SMDP methods**

# Outline

- The RL (MDP) framework
- The extension to temporally abstract “options”
  - ❖ Options and Semi-MDPs
  - ❖ Hierarchical planning and learning
- Rooms example
- Between MDPs and Semi-MDPs
  - ❖ Improvement by interruption (including Spy plane demo)
  - ❖ A taste of
    - Intra-option learning
    - Subgoals for learning options
    - RoboCup soccer demo



# Tasks and Subgoals

It is natural to define **options as solutions to subtasks**  
e.g. **treat hallways as subgoals, learn shortest paths**

We have defined subgoals as pairs :  $\langle G, g \rangle$

$G \subseteq S$  is the set of states treated as subgoals

$g: G \mapsto \mathfrak{R}$  are their subgoal values (can be both good and bad)

Each subgoal has its own set of value functions, e.g. :

$$V_g^\mu(s) = E \{ r_1 + \gamma r_2 + \dots + \gamma^{k-1} r_k + g(s_k) \mid s_0 = s, \mu \}$$

$$V_g^*(s) = \max_{\mu} V_g^\mu(s)$$

First state in G

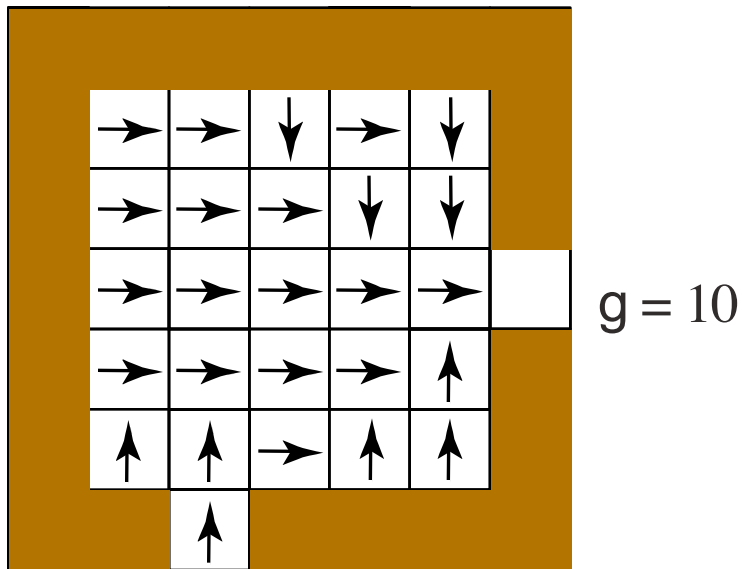


Policies inside options can be learned from subgoals,  
in intra - option, off - policy manner.

# Options Depend on Outcome Values

Small negative rewards on each step

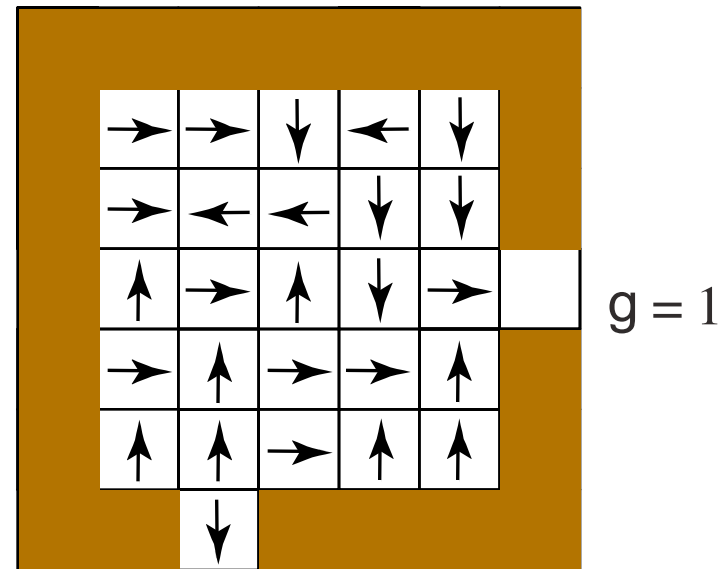
Large Outcome Values



$g = 0$

Learned Policy: Shortest Paths

Small Outcome Values

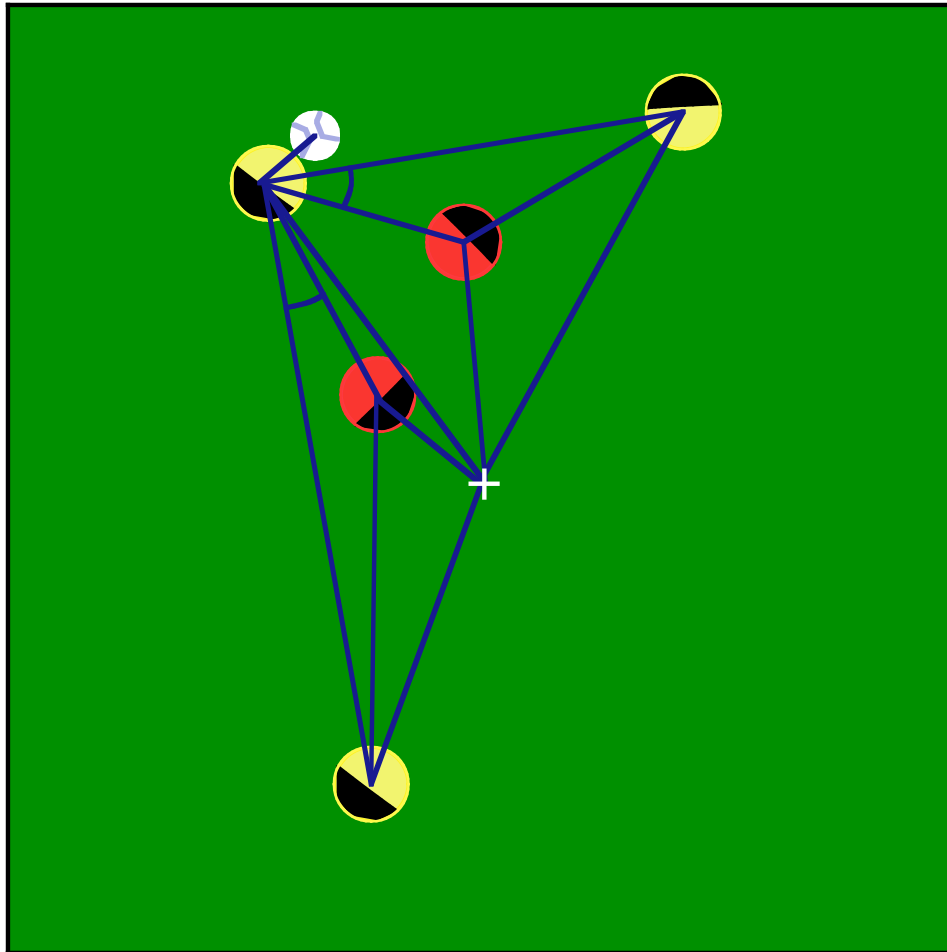


$g = 0$

Learned Policy: Avoids Negative Rewards

# 13 Continuous State Variables (for 3 vs 2)

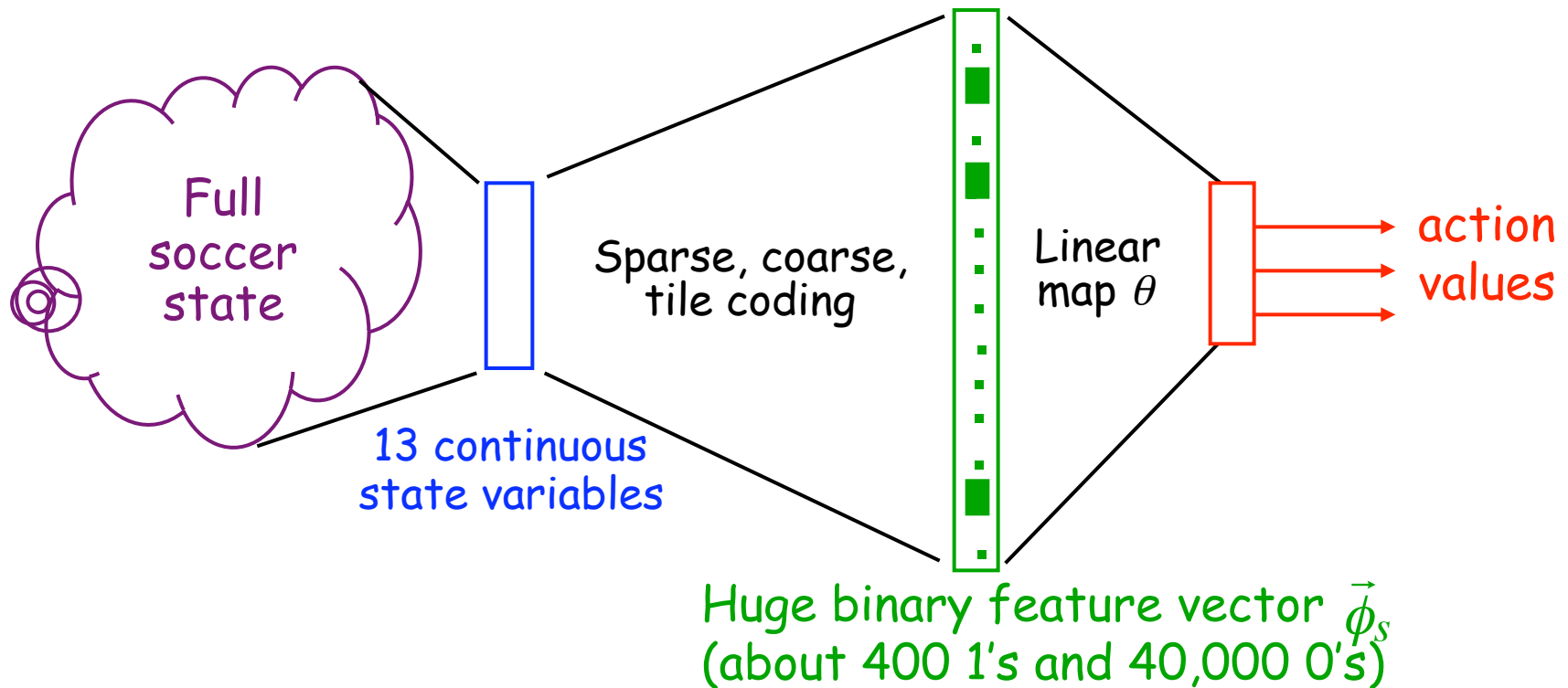
Browser demo



11 distances among  
the players, ball,  
and the center of  
the field

2 angles to takers  
along passing lanes

# RoboCup Feature Vectors



# **RoboCup Demo**

From Stone, Sutton, Kuhlmann, 2005

# Summary: Benefits of Options

- **Transfer of knowledge**
  - ❖ Solutions to sub-tasks can be saved and reused
  - ❖ Domain knowledge can be provided as options and subgoals
- **Potentially much faster** learning and planning
  - ❖ By representing action at an appropriate temporal scale
- Models of options are a form of **knowledge representation**
  - ❖ Expressive
  - ❖ Clear
  - ❖ Suitable for learning and planning
- **Much more to learn** than just one policy, one set of values
  - ❖ A framework for “**constructivism**” – for finding models of the world that are useful for rapid planning and learning

# Conclusions

- We have come a long way toward linking human-level choices to microscopic actions
  - ❖ Temporally abstract facts, and estimates of them - knowledge!
  - ❖ A theory of how to combine known subcontrollers (behaviors)
  - ❖ Beginnings of how to learn them efficiently and without interference
    - Resolution of the “subgoal credit-assignment” problem
- We have shown how the high-level can mirror the low
  - ❖ It’s all choices, states, and values
  - ❖ A minimal extension of existing RL/MDP ideas
- The state assumption remains a problem
  - ❖ Someday options may revolutionize our notion of state and of perception
  - ❖ By combining with ideas of *predictive state representations*