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

Options – Temporally Abstract Actions

An option is a triple, $o = \langle \pi_o, \gamma_o \rangle$

$\pi_o : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the policy followed during o

$\gamma_o : \mathcal{S} \rightarrow [0, \gamma]$ is the probability of the option continuing (not terminating) in each state

Execution is nominally hierarchical (call-and-return)

E.g., the `docking` option:

π_o: hand-crafted controller

γ_o: 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, $\mathcal{A}(s)$
It also has a set of options, $\mathcal{O}(s)$

Just as we can have a flat policy, over actions, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$
We can have a hierarchical policy, over *options*, $h : \mathcal{O} \times \mathcal{S} \rightarrow [0,1]$

To execute h in s :

select option o with probability $h(o|s)$
follow o until it terminates, in s'
then choose a next option with probability $h(o'|s')$ again, and so on

Every hierarchical policy determines a flat policy

$$\pi = f(h)$$

Even if all the options are Markov, $f(h)$ is usually not Markov

Actions are a special case of options

Value Functions with Temporal Abstraction

Define value functions for hierarchical policies and options:

$$v_h(s) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \;\middle|\; S_t = s, A_{t:\infty} \sim h\right]$$

$$q_h(s,o) = \mathbb{E}[G_t \mid S_t = s, A_{t:t+k-1} \sim \pi_o, k \sim \gamma_o, A_{t+k:\infty} \sim h]$$

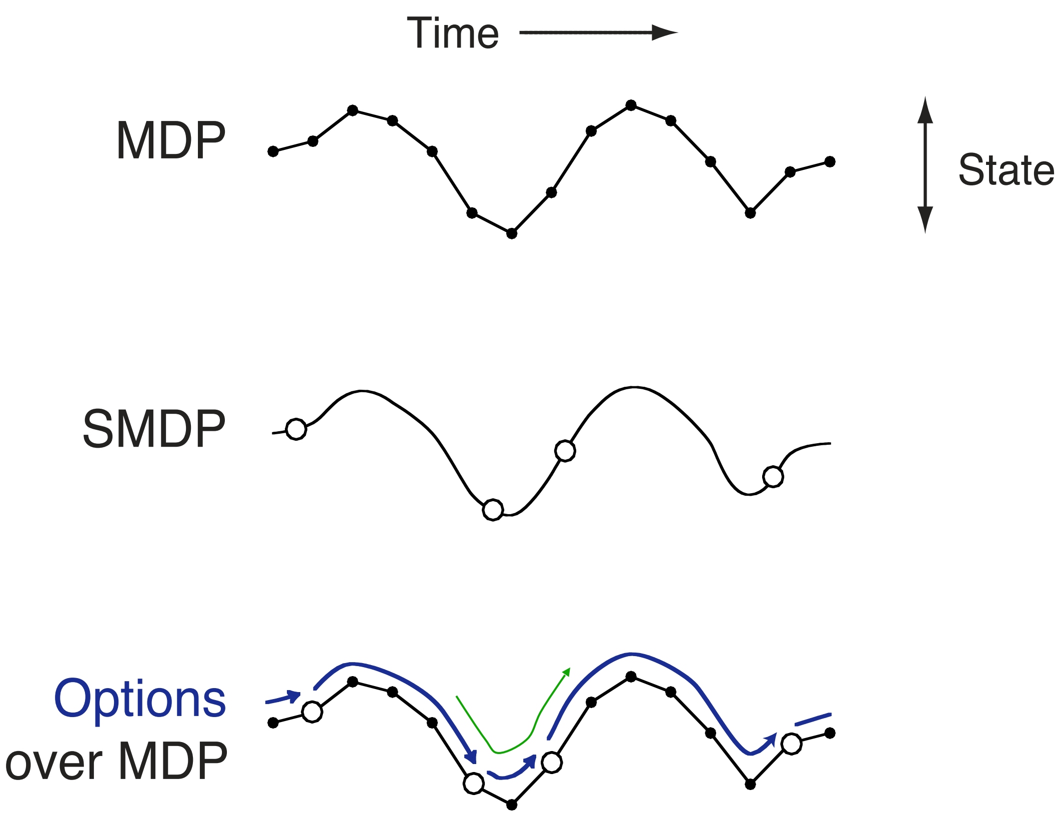
Now consider a limited set of options $\mathcal{O}$
and hierarchical policies that choose only from them $h \in \Pi(\mathcal{O})$

$$v_*^{\mathcal{O}}(s) = \max_{h \in \Pi(\mathcal{O})} v_h(s)$$

A new set of optimization problems

$$q_*^{\mathcal{O}}(s,o) = \max_{h \in \Pi(\mathcal{O})} q_h(s,o)$$

Options define a Semi-Markov Decision Process (SMDP) overlaid on the MDP



Discrete time
Homogeneous discount

Continuous time
Discrete events
Interval-dependent discount

Discrete time
Overlaid discrete events
Interval-dependent discount

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

Models of the Environment with Temporal Abstraction

Planning requires models of the consequences of action

The model of an action has a reward part and a state transition part:

$$r(s,a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$
$$p(s'|s,a) = \Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\}$$

As does the model of an option:

$$r(s,o) = \mathbb{E}\left[R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} \;\middle|\; S_t = s, A_{t:t+k-1} \sim \pi_o, k \sim \gamma_o\right]$$

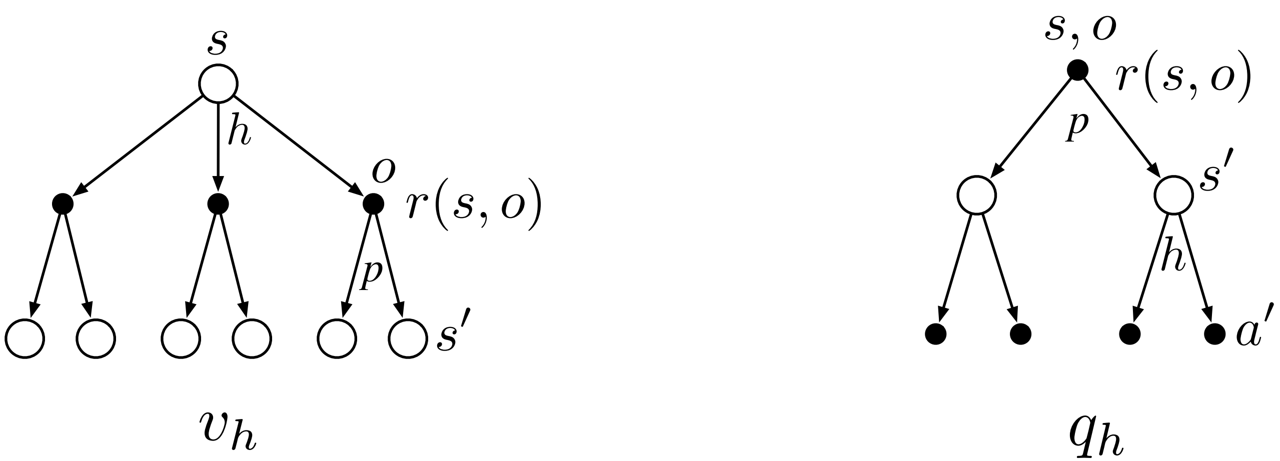
$$p(s'|s,o) = \sum_{k=1}^{\infty} \Pr\{S_{t+k} = s', \text{termination at } t+k \mid S_t = s, A_{t:t+k-1} \sim \pi_o\}\, \gamma^k$$

Bellman Equations with Temporal Abstraction

For policy-specific value functions:

$$v_h(s) = \sum_o h(o|s) \left[r(s,o) + \sum_{s'} p(s'|s,o) v_h(s') \right]$$

$$q_h(s,o) = r(s,o) + \sum_{s'} p(s'|s,o) \sum_{o'} h(o'|s') q_h(s',o')$$



Planning with Temporal Abstraction

Initialize: $V(s) \leftarrow 0, \qquad \forall s \in \mathcal{S}$

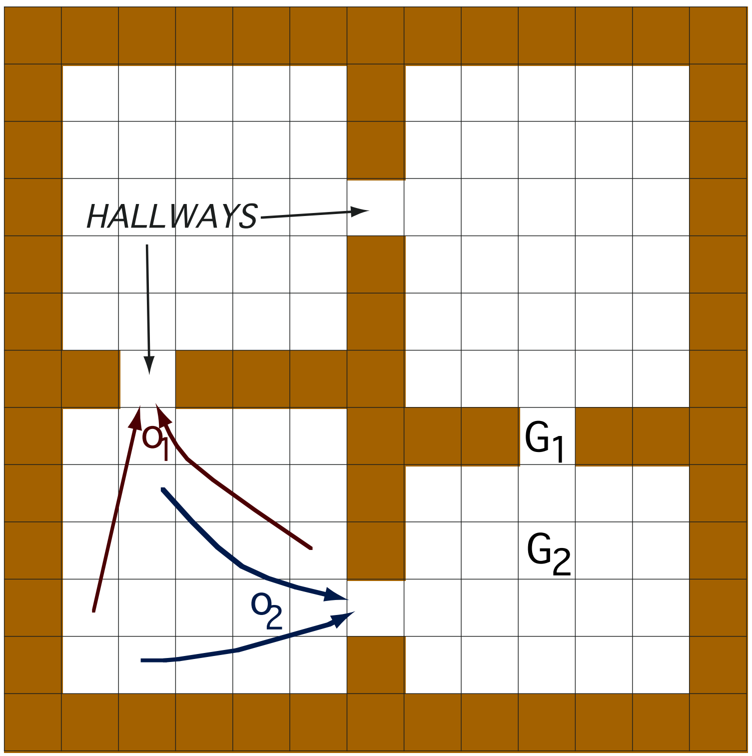
Iterate: $V(s) \leftarrow \max_o \left[r(s,o) + \sum_{s'} p(s'|s,o) V(s') \right]$

$$V \rightarrow v_*^{\mathcal{O}}$$

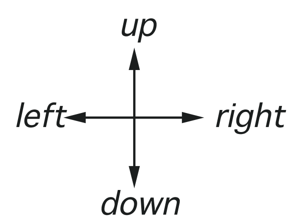
$$h_*^{\mathcal{O}}(s) = \text{greedy}(s, v_*^{\mathcal{O}}) = \arg\max_{o \in \mathcal{O}} \left[r(s,o) + \sum_{s'} p(s'|s,o) v_*^{\mathcal{O}}(s') \right]$$

Reduces to conventional value iteration if $\mathcal{O} = \mathcal{A}$

Rooms Example



4 stochastic primitive actions



Fail 33% of the time

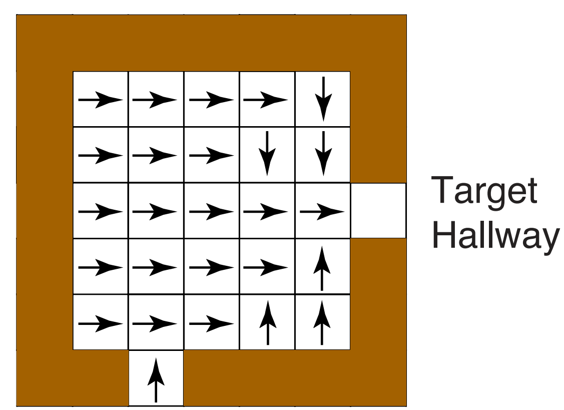
8 multi-step options

(to each room's 2 hallways)

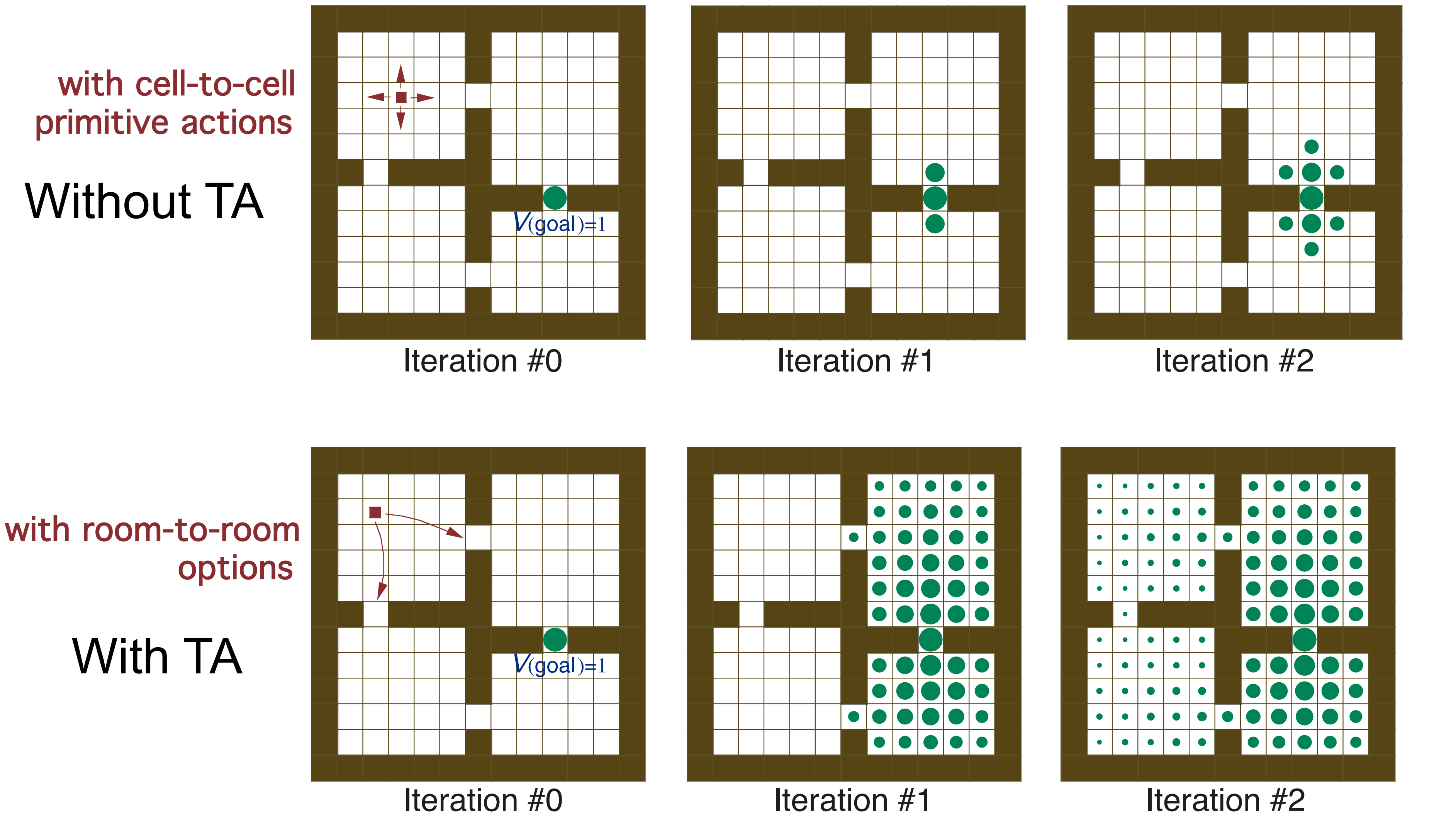
All rewards zero,
except +1 into goal

$\gamma = .9$

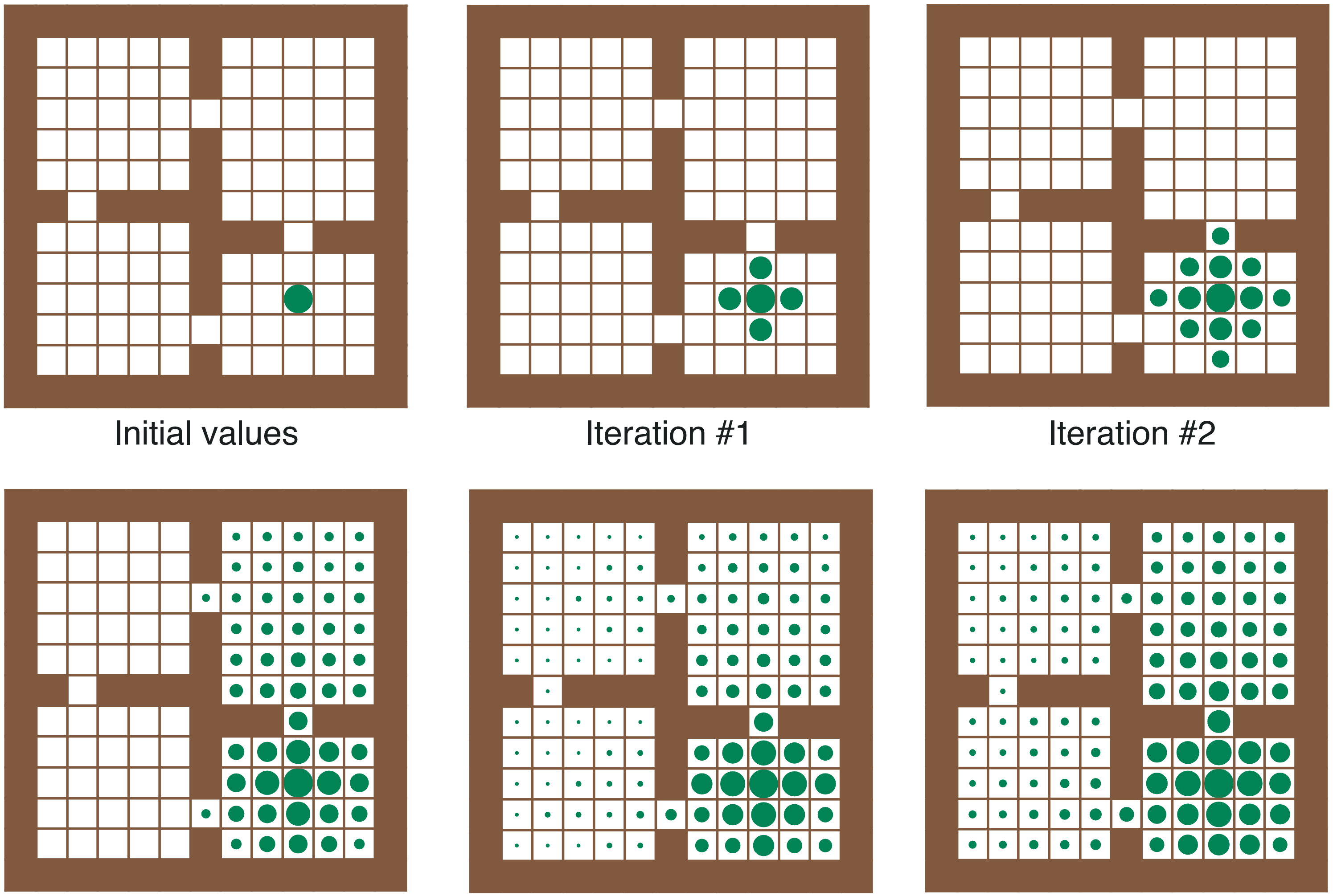
Policy of one option:



Planning is much faster with Temporal Abstraction



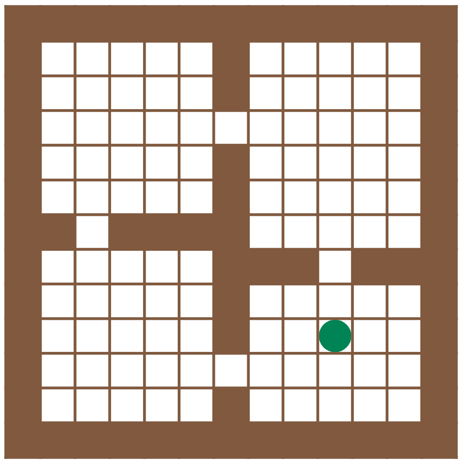
Temporal Abstraction helps even with Goal≠Subgoal given both primitive actions and options



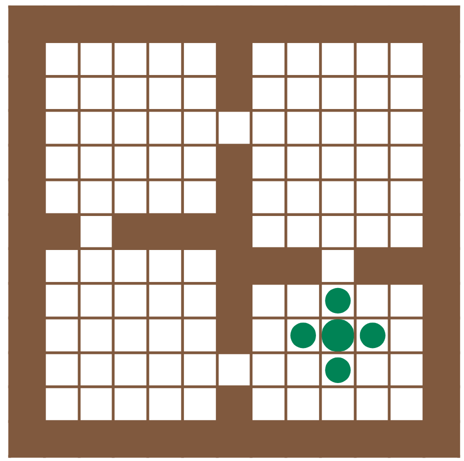
Initial values Iteration #1 Iteration #2

Iteration #3 Iteration #4 Iteration #5

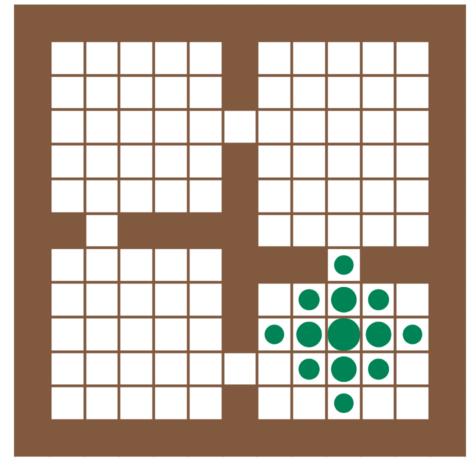
Temporal Abstraction helps even with Goal≠Subgoal given both primitive actions and options



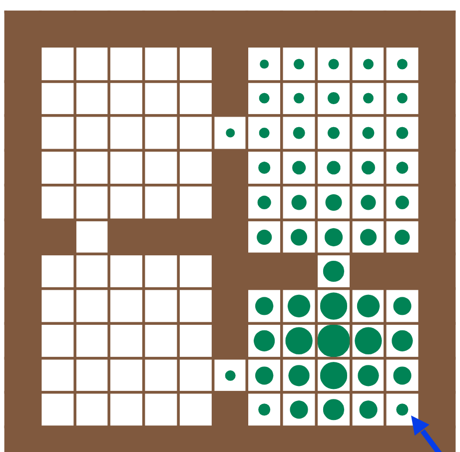
Initial values



Iteration #1

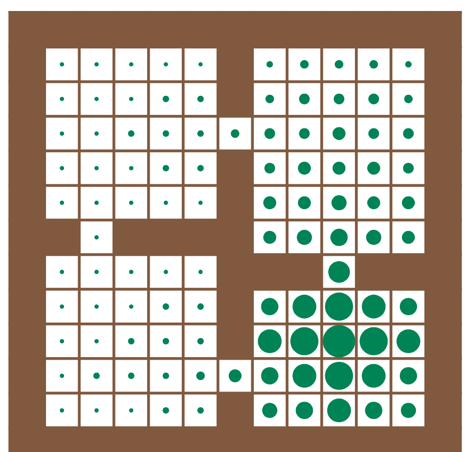


Iteration #2

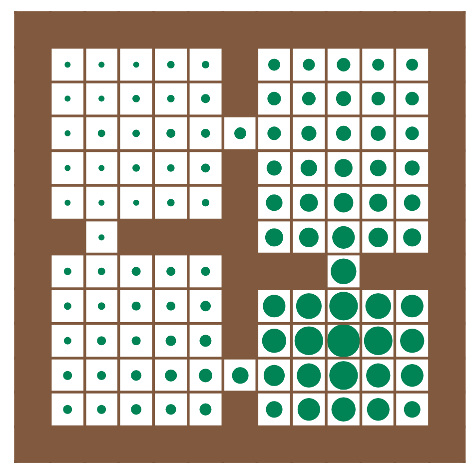


Iteration #3

why?

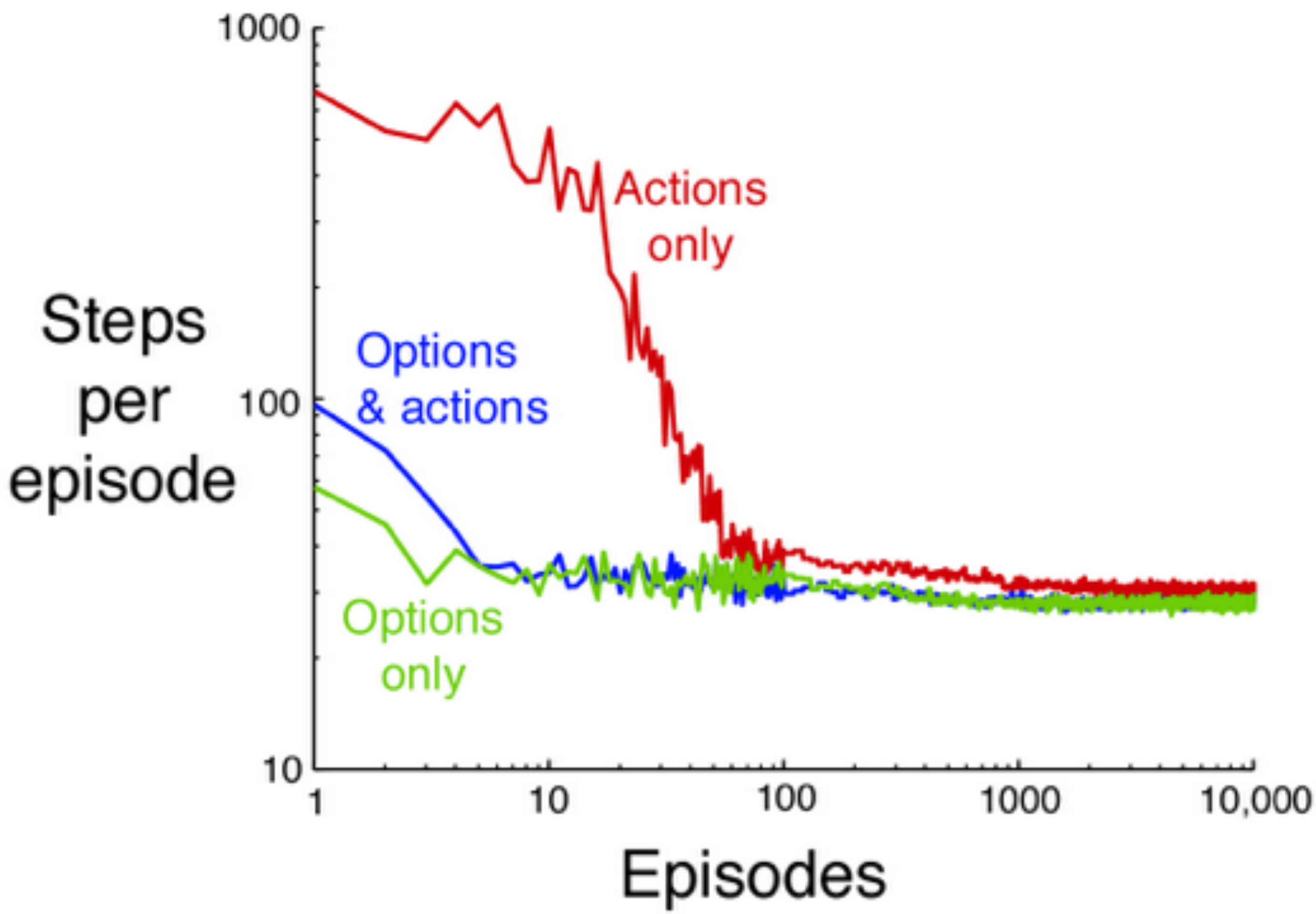


Iteration #4



Iteration #5

Temporal abstraction also speeds learning about path-to-goal



SMDP Theory Provides a lot of this

- Hierarchical policies over options: $h(o|s)$
- Value functions over options: $v_h(s), q_h(s,o), v_*^{\mathcal{O}}(s), q_*^{\mathcal{O}}(s,o)$
- Learning methods: Bradtke & Duff (1995), Parr (1998)
- Models of options: $r(s,o), p(s'|s,o)$
- 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 not all.
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 γ_o says to

Theorem: For any hierarchical policy $h : \mathcal{O} \times \mathcal{S} \to [0,1]$,
suppose we interrupt its options one or more times, t,
when the action we are about to take o, is such that

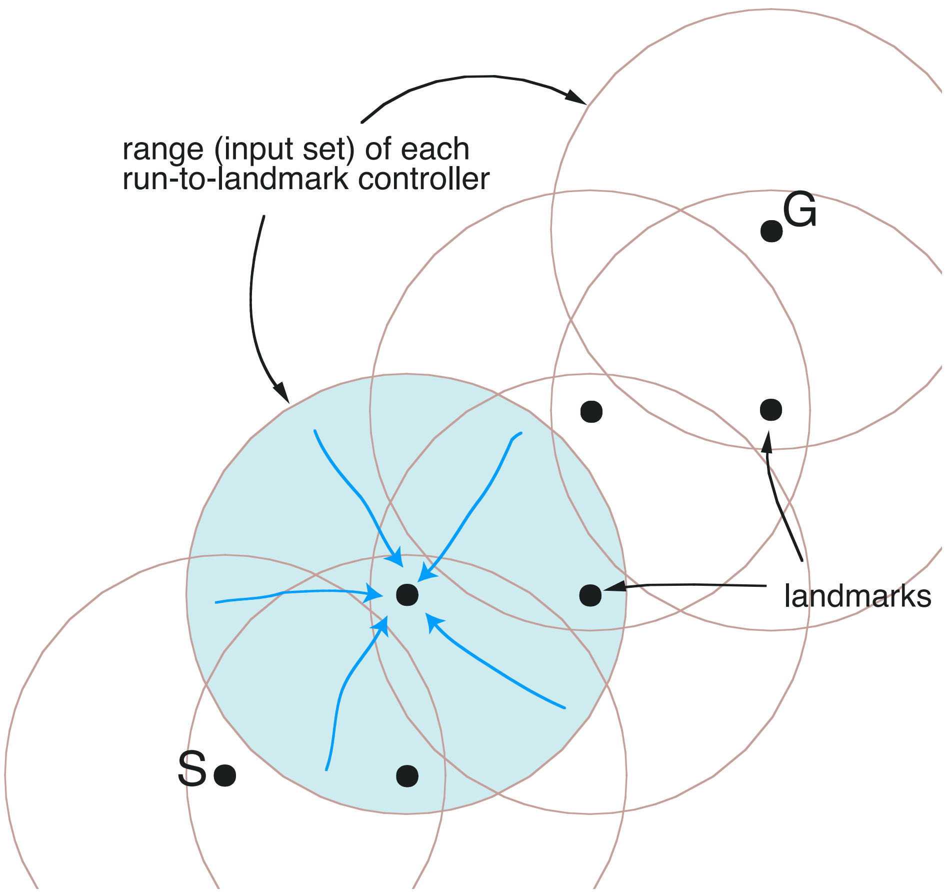
$$q_h(S_t, o) < q_h(S_t, h(S_t))$$

to obtain h',

Then $h' \geq h$ (it attains more or equal reward everywhere)

Application: Suppose we have determined $q_*^{\mathcal{O}}$ and thus $h = h_*^{\mathcal{O}}$
Then h' is guaranteed better than $h_*^{\mathcal{O}}$
and is available with no further computation

Landmarks Task



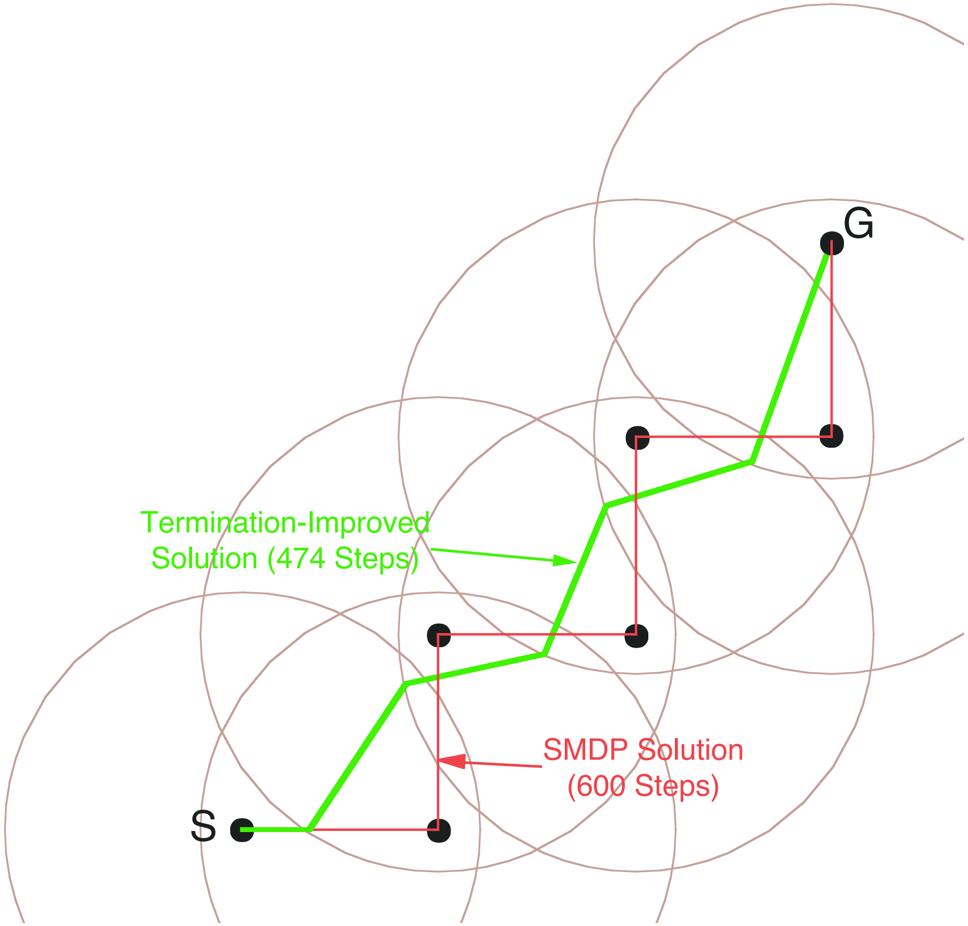
Task: navigate from S to G as fast as possible

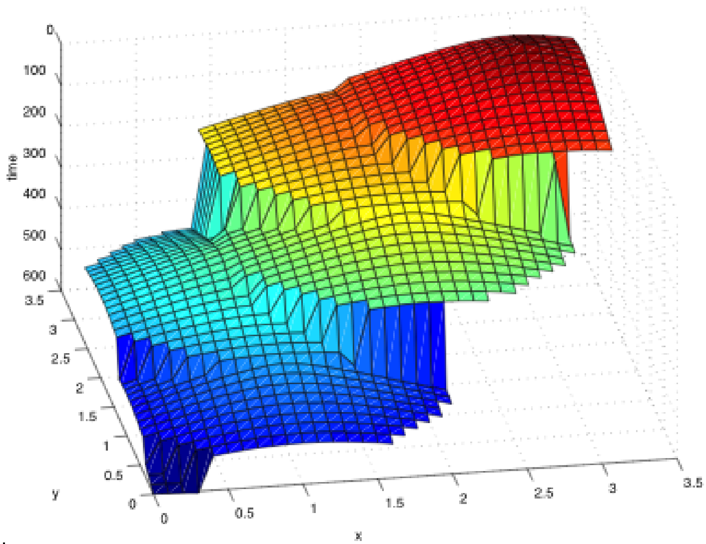
4 primitive actions, for taking tiny steps up, down, left, right

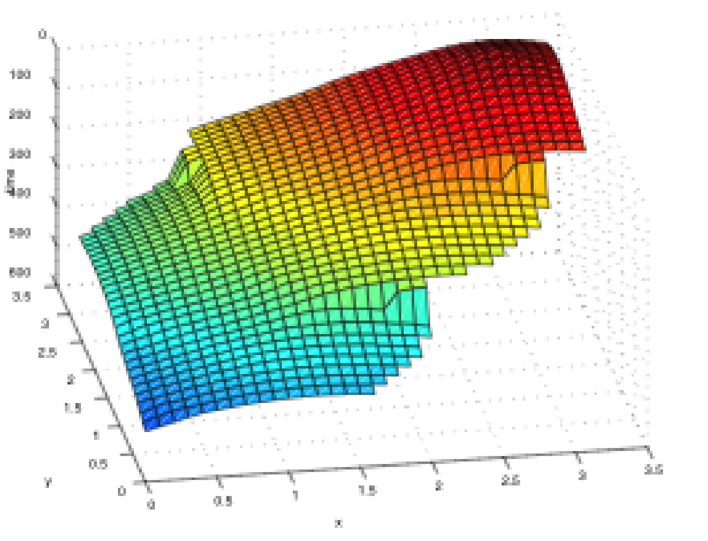
7 controllers for going straight to each one of the landmarks, from within a circular region where the landmark is visible

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

Termination Improvement for Landmarks Task

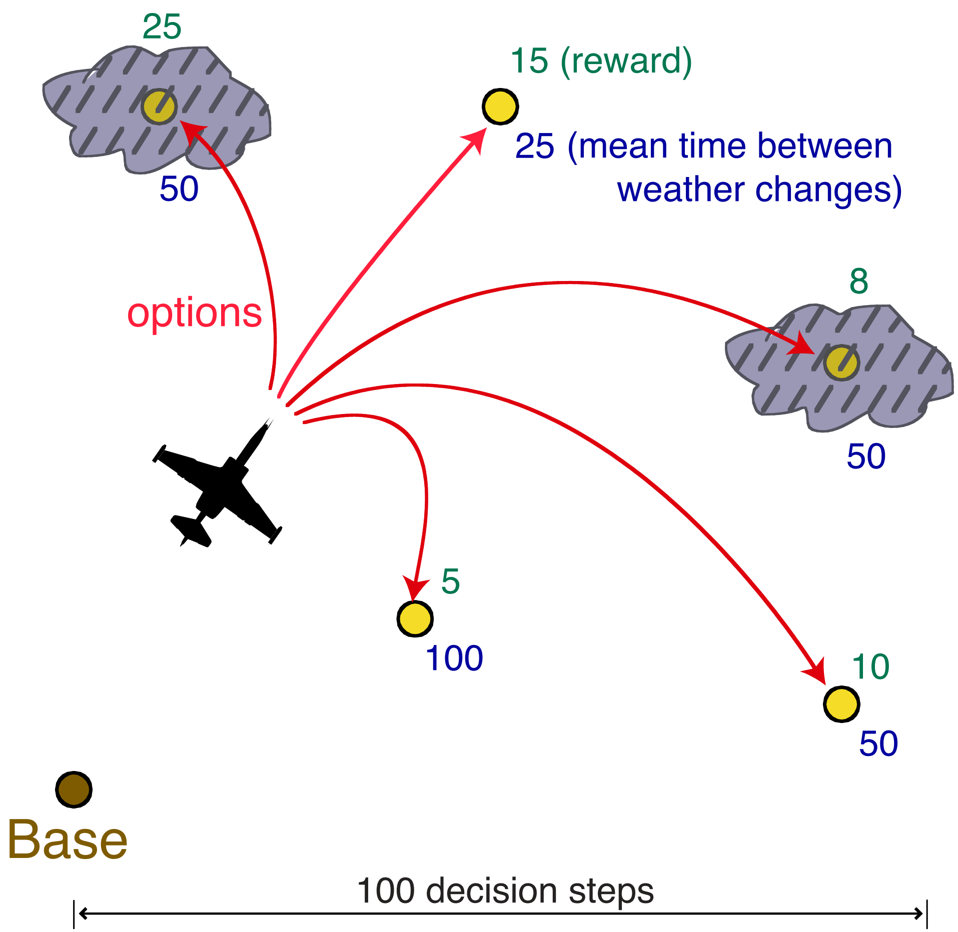






Allowing early termination based on models improves the value function at no additional cost!

Spy Plane Example



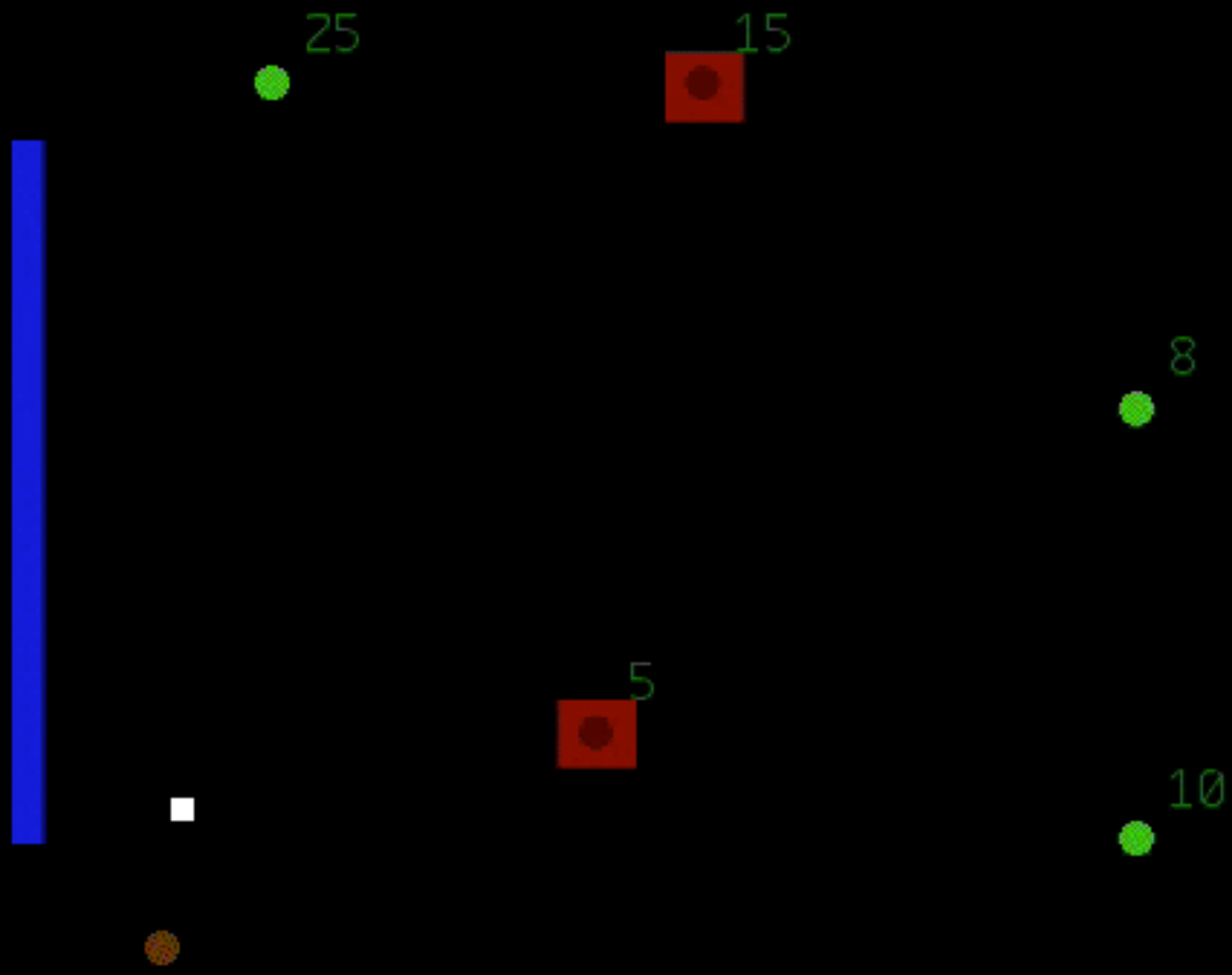
- Mission: Fly over (observe) most valuable sites and return to base
- Stochastic weather affects observability (cloudy or clear) of sites
- Limited fuel
- Intractable with classical optimal control methods
- Temporal scales:
  - Actions: which direction to fly now
  - Options: which site to head for
- Options compress space and time
  - Reduce steps from ~600 to ~6
  - Reduce states from ~10^{10} to ~10^6

$$q_*^{\mathcal{O}}(s,o) = r(s,o) + \sum_{s'} p(s'|s,o) v_*^{\mathcal{O}}(s')$$

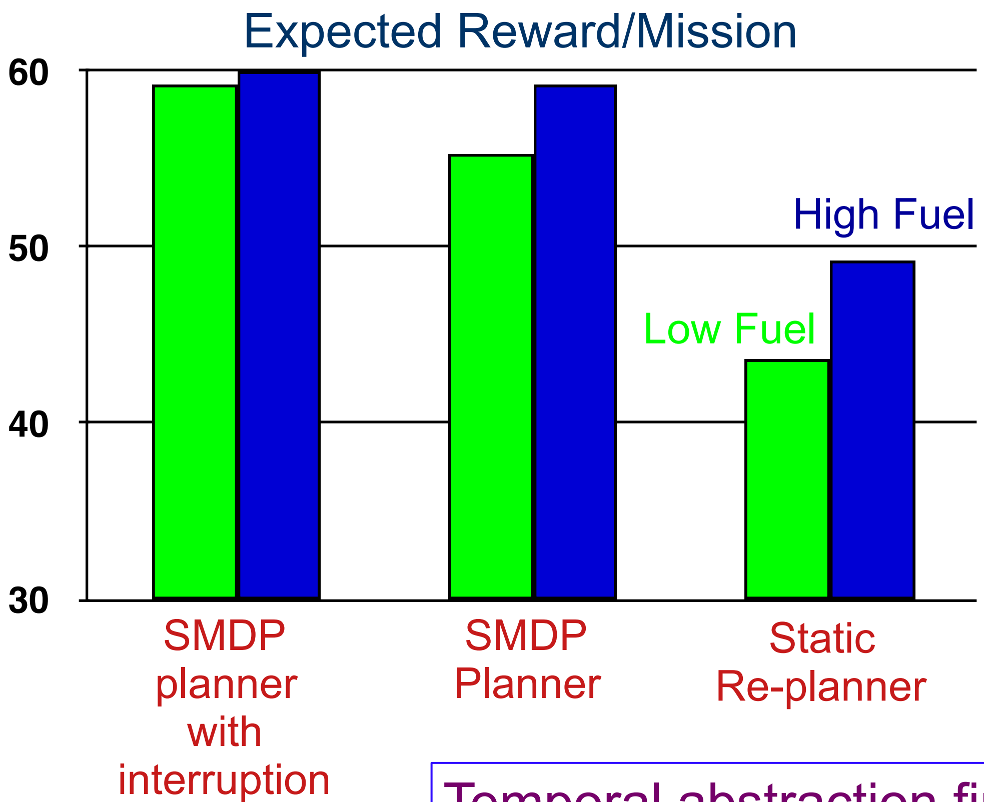
any state ~10^{10}

sites only ~10^6

Spy Drone



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

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

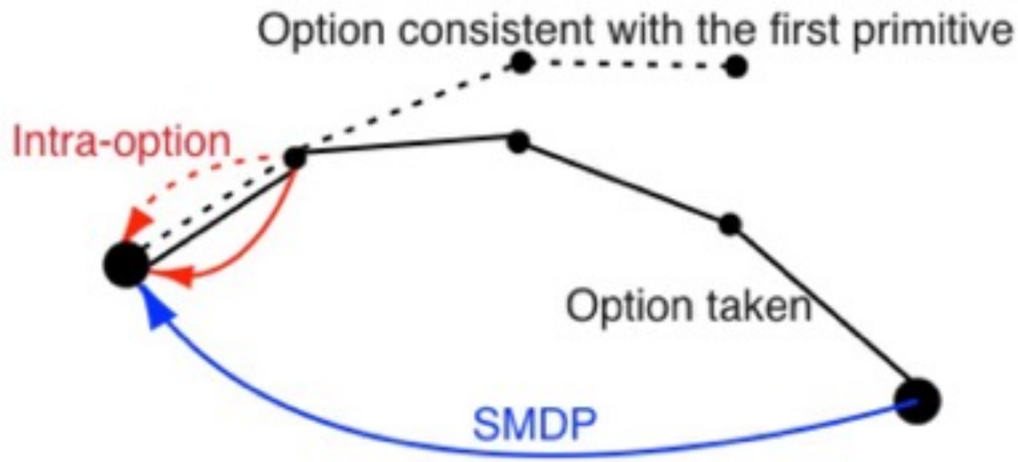
Intra-option Q-learning:

- 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

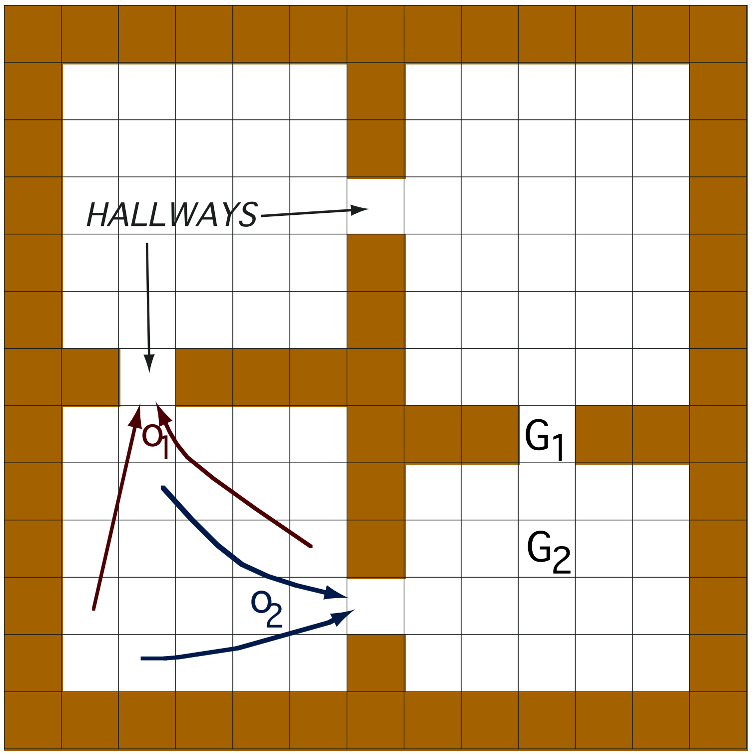


SMDP Learning: <u>execute option to termination</u>,then update only the option taken

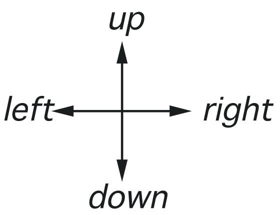
Intra-Option Learning: <u>after each primitive action</u>, update all the options that could have taken that action

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

Returning to the rooms example...



4 stochastic primitive actions



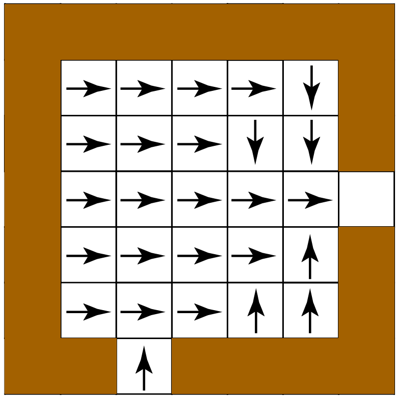
8 multi-step options

(to each room's 2 hallways)

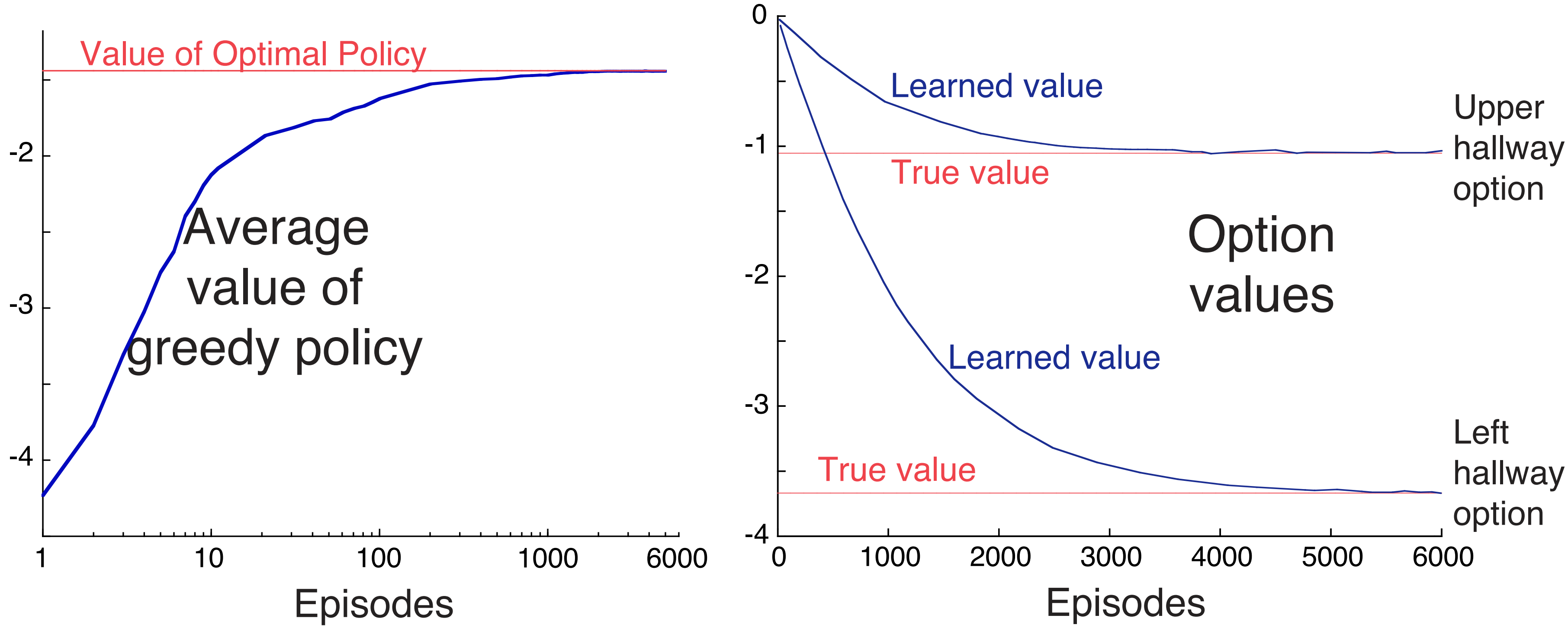
All rewards zero, except +1 into goal

$\gamma = .9$

Policy of one option:



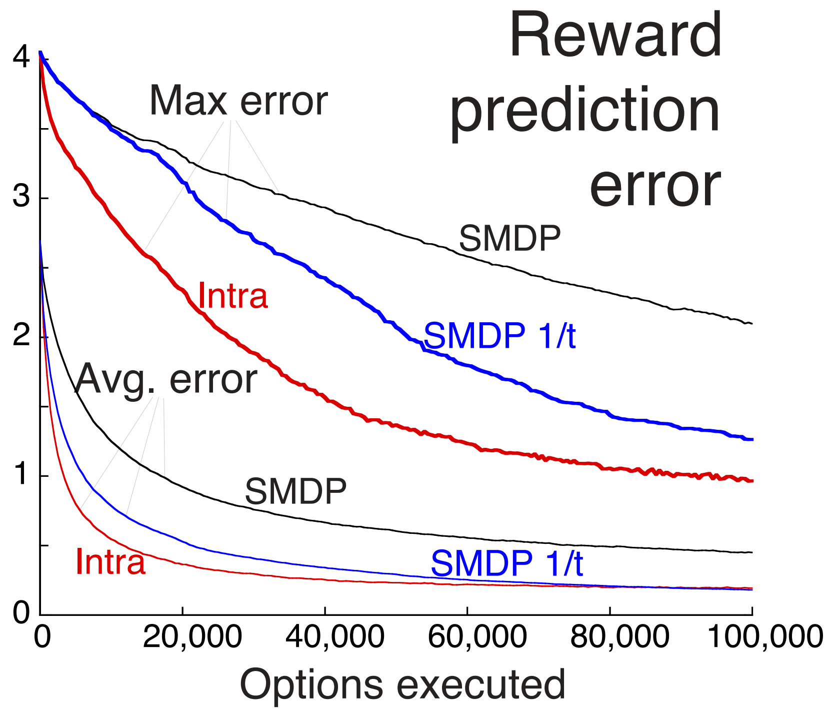
Intra-Option Value Learning in the Rooms Example

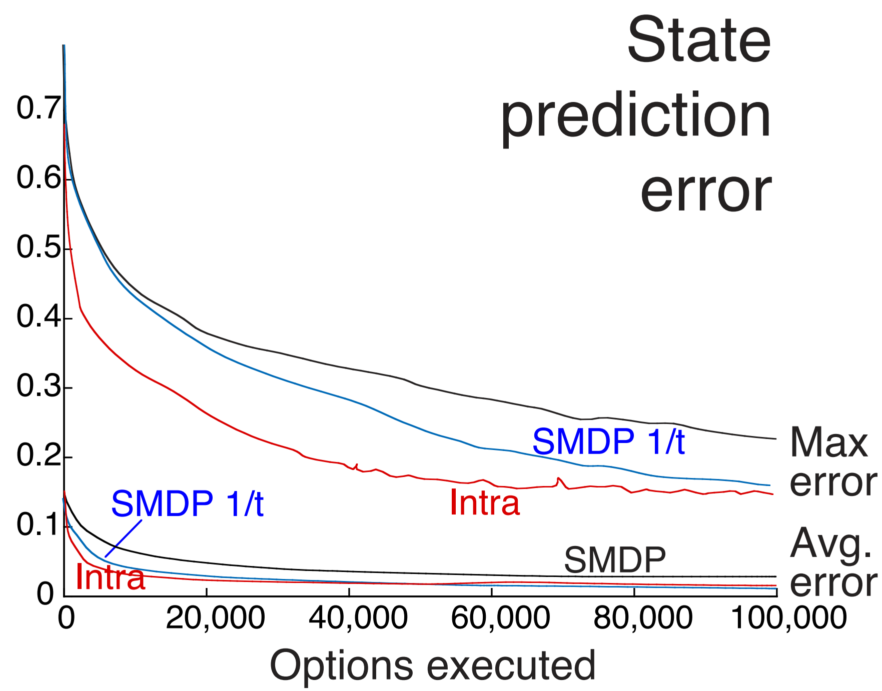


Random start, goal in right hallway, random actions

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

Intra-Option Model Learning





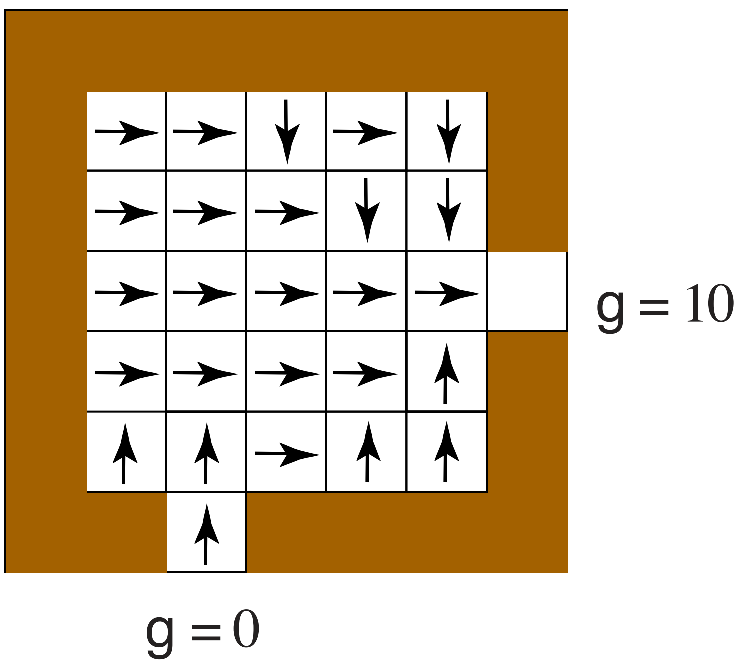
Random start state, no goal, pick randomly among all options

Intra-option methods work much faster than SMDP methods

Options Depend on Outcome Values

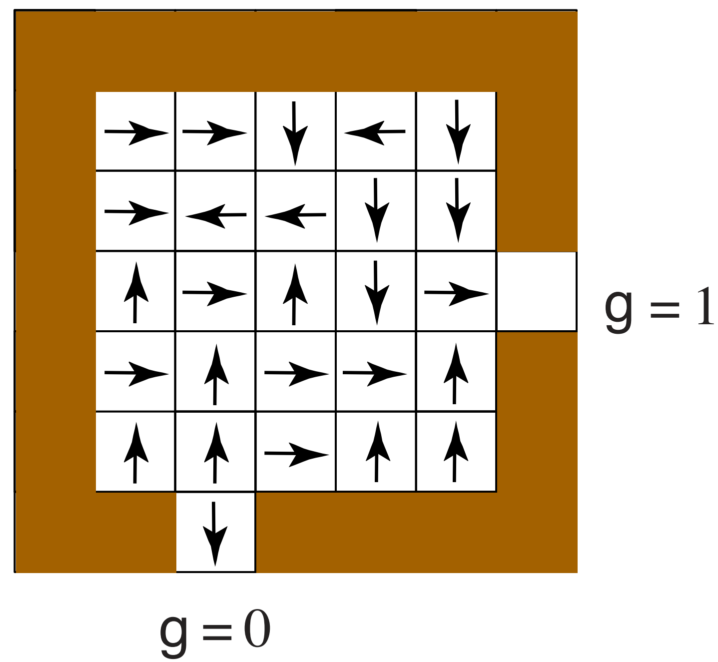
Small negative rewards on each step

Large Outcome Values



Learned Policy: Shortest Paths

Small Outcome Values



Learned Policy: Avoids Negative Rewards

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" or "continual learning" – 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