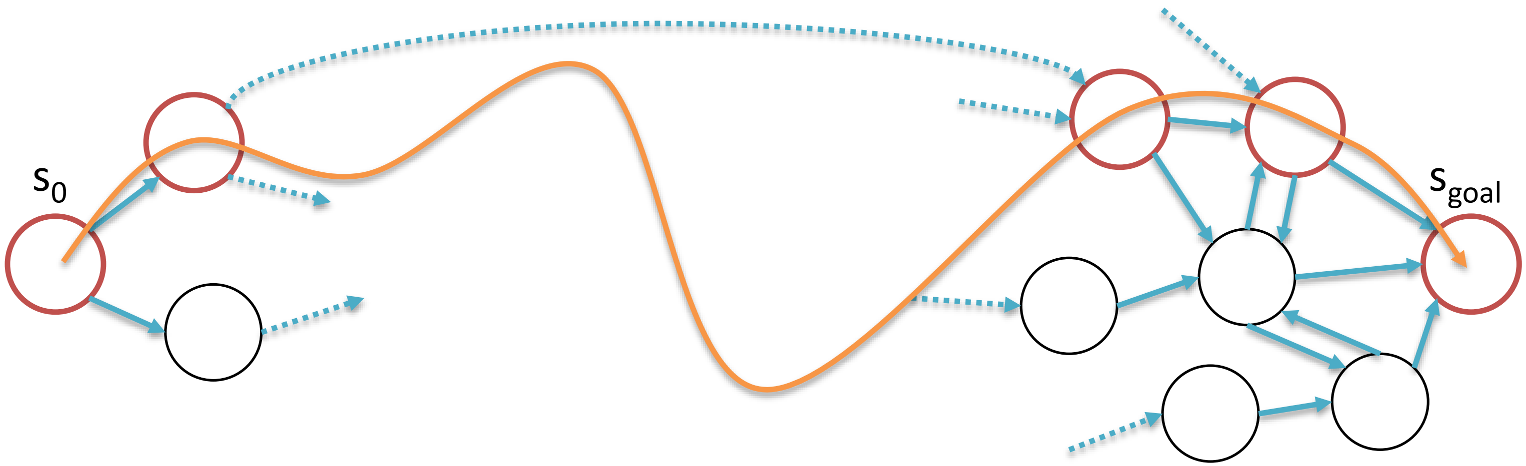
Probabilistic Planning

Acting when not everything is under your control



16.410/413 Principles of Autonomy and Decision-Making

Pedro Santana (psantana@mit.edu)

October 14th, 2015.

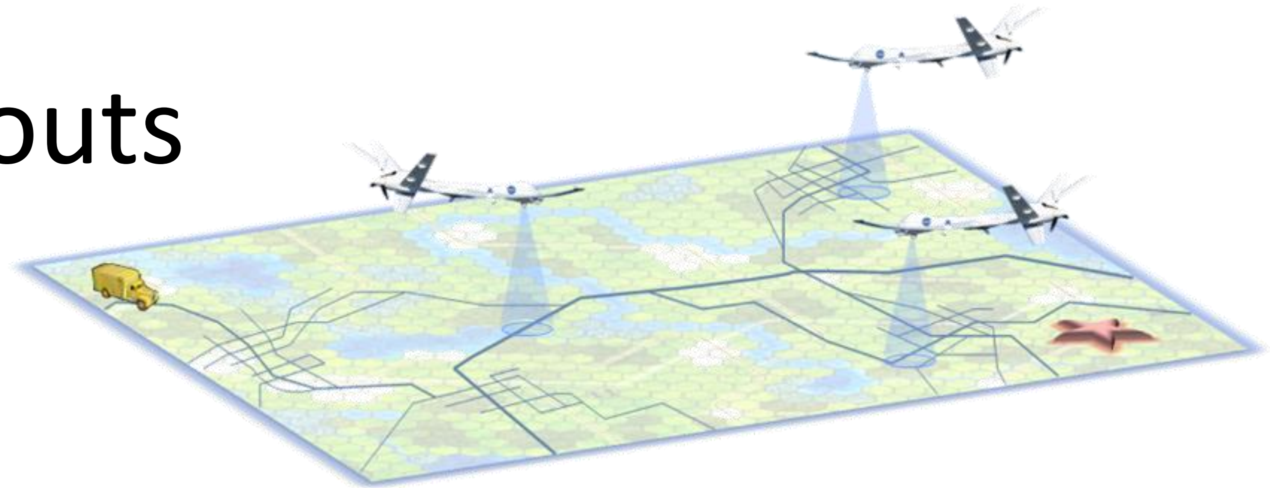
Assignments

- Problem set 5
  - Out last Wednesday.
  - Due at midnight **this Friday**.
- Midterm on Monday, October 19th. Good luck, even though you won't need it! ;)
- Readings
  - "Beyond Classical Search" [AIMA], Sections 4.3 and 4.4;
  - "Quantifying Uncertainty" [AIMA], Ch. 13;
  - "Making Complex Decisions" [AIMA], Ch. 17;
  - (Optional) Kolobov & Mausam, "Probabilistic Planning with Markov Decision Processes", Tutorial.
  - (Optional) Hansen & Zilberstein, "LAO*: a heuristic search algorithm that finds solutions with loops", AI, 2001.

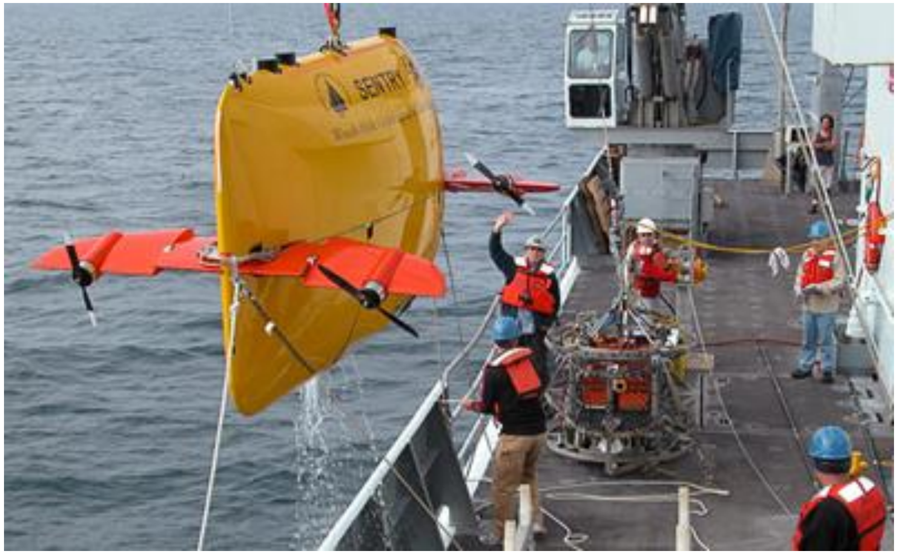
1. Motivation

Where can probabilistic planning be useful?

Science scouts

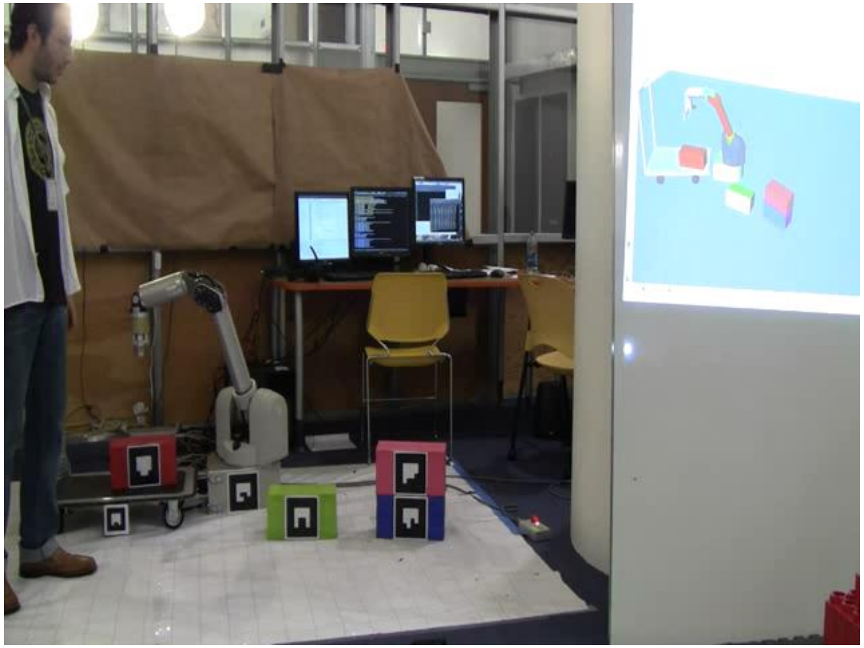


Courtesy of Andrew J. Wang

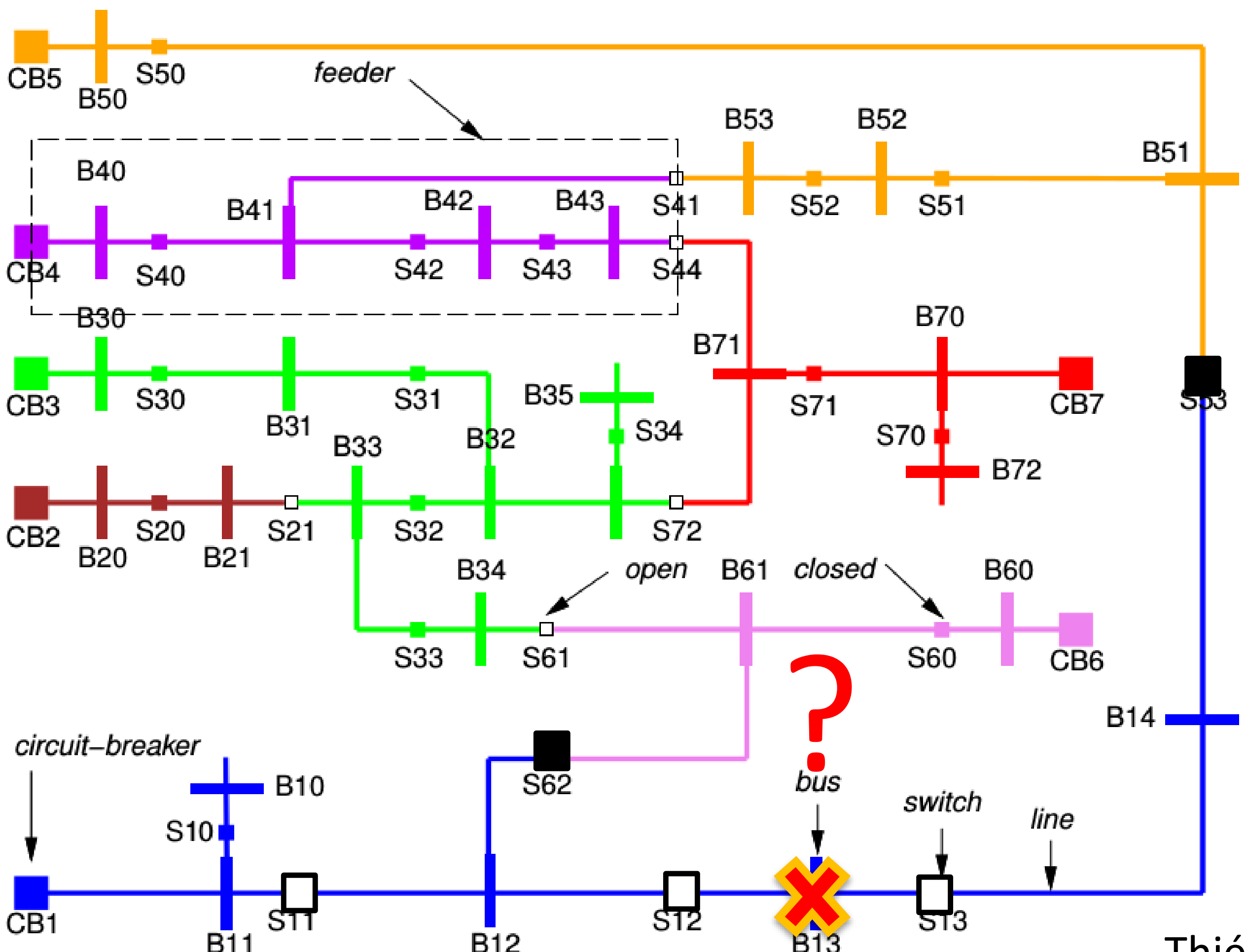


MIT AEROASTRO

Collaborative manufacturing



Power supply restoration



Thiébaux & Cordier (2001)

How this relates to what we've seen?

L06: Activity Planning
David Wang

Planning as state-space search.

L08: Markov Decision Processes
Brian Williams

Plans that optimize utility

L11: Probabilistic Planning

Observe-Act as AND-OR search.

Markovian hidden state inference.

Computing utility with probabilistic transitions.

L10: Adversarial Games
Tiago Vaquero

L09: Hidden Markov Models
Pedro Santana

10/09/15 Recitation
Enrique Fernández

Our goal for today

How can we generate ***plans*** *that* ***optimize performance*** *when controlling a system with* ***stochastic transitions*** *and* ***hidden state****?*

Today’s topics

1. Motivation
2. MDP recapitulation
3. Search-based probabilistic planning
4. AO*
5. LAO*
6. Extension to hidden states

2. MDP recapitulation

Where we will:

- *recall what we've learned about MDPs;*
- *learn that recap = recapitulation.*

Elements of an MDP

$\mathbb{S}$: discrete set of states.

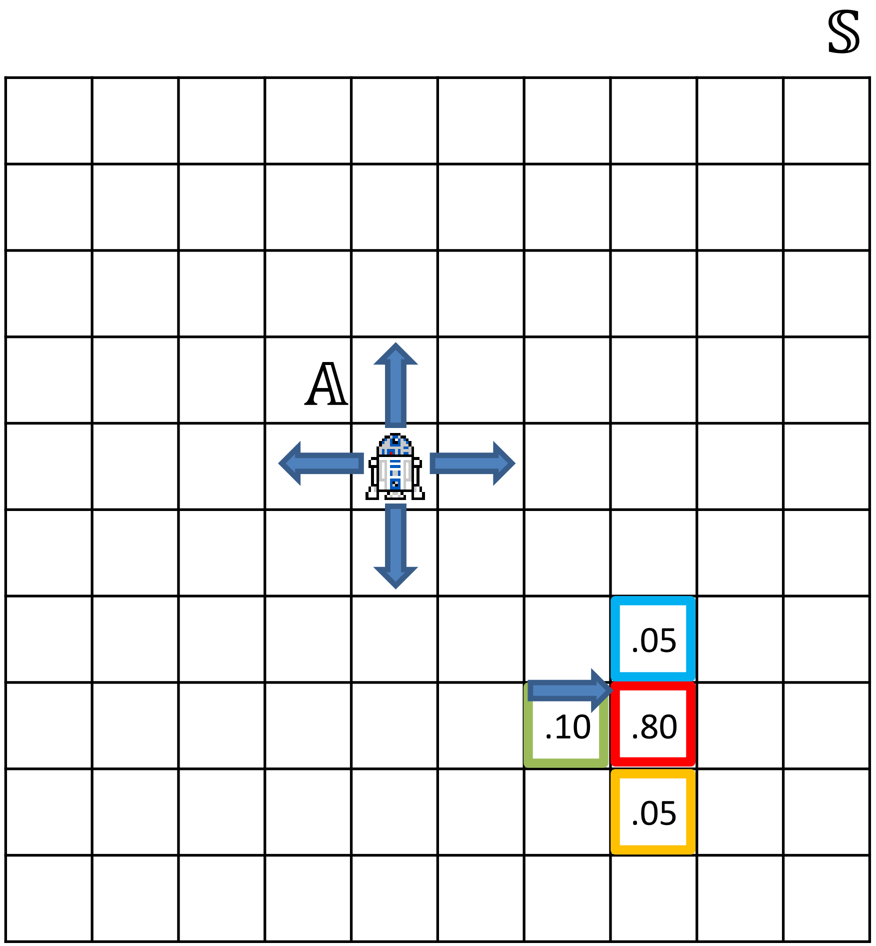
$\mathbb{A}$: discrete set of actions.

T: $\mathbb{S}\times\mathbb{A}\times\mathbb{S}\rightarrow [0,1]$, *transition function*

R: $\mathbb{S}\times\mathbb{A}\rightarrow\mathbb{R}$, *reward function.*

$\gamma \in [0,1]$: discount factor

$$T(s_k, a_k, s_{k+1}) = \Pr(s_{k+1}|s_k, a_k)$$



Solving MDPs through VI

Unknown

$$\underbrace{V(s_k, a_k)}_{\text{Expected reward of executing } a_k \text{ at } s_k} = \underbrace{R(s_k, a_k)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s_{k+1}} \underbrace{V^*(s_{k+1})}_{\text{Optimal reward at } s_{k+1}} T(s_k, a_k, s_{k+1})}_{\text{Expected, discounted optimal future reward}}$$

Value iteration

$$V^{*(0)}(s_k), \forall s_k$$

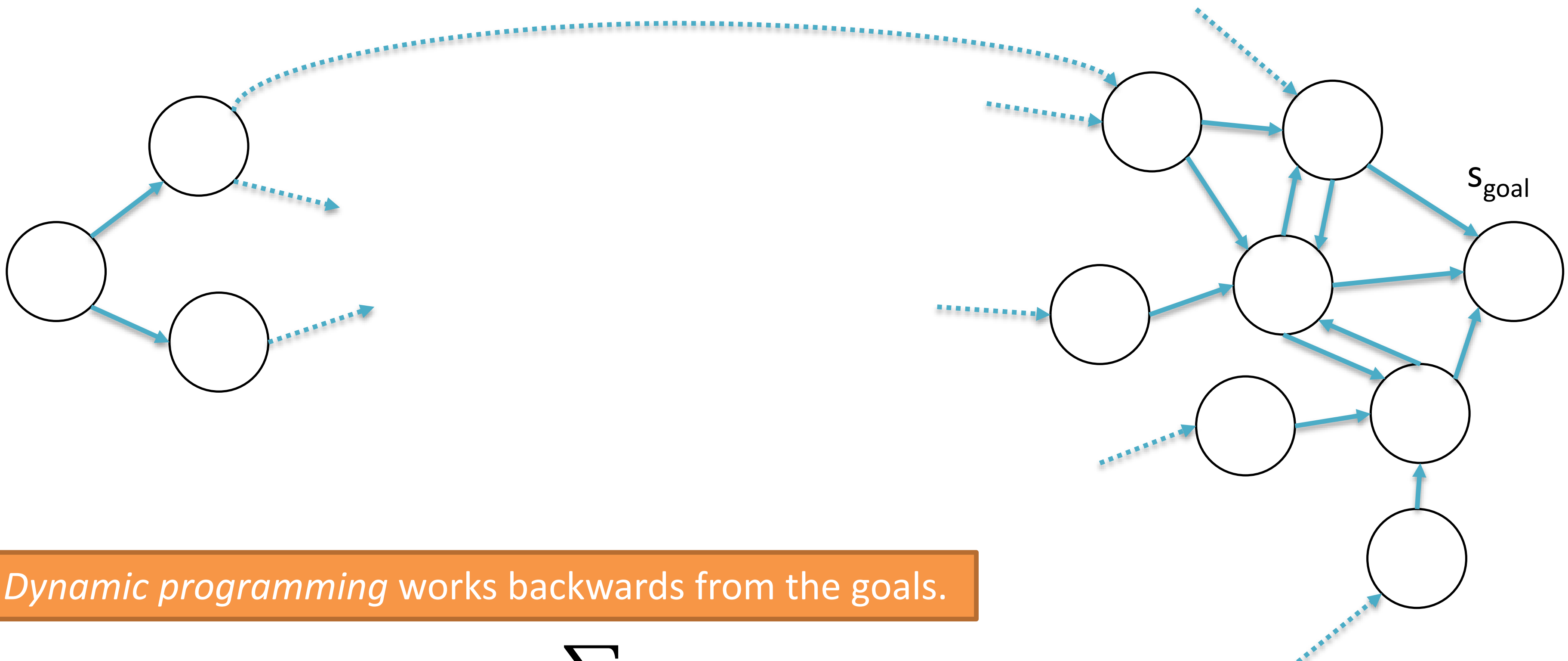
After convergence

$$\pi^*(s) = \arg\max_a V(s, a)$$

$$V^{*(t+1)}(s_k) = \max_{a_k} R(s_k, a_k) + \gamma \sum_{s_{k+1}} V^{*(t)}(s_{k+1})\, T(s_k, a_k, s_{k+1})$$

[AIMA] Section 17.2

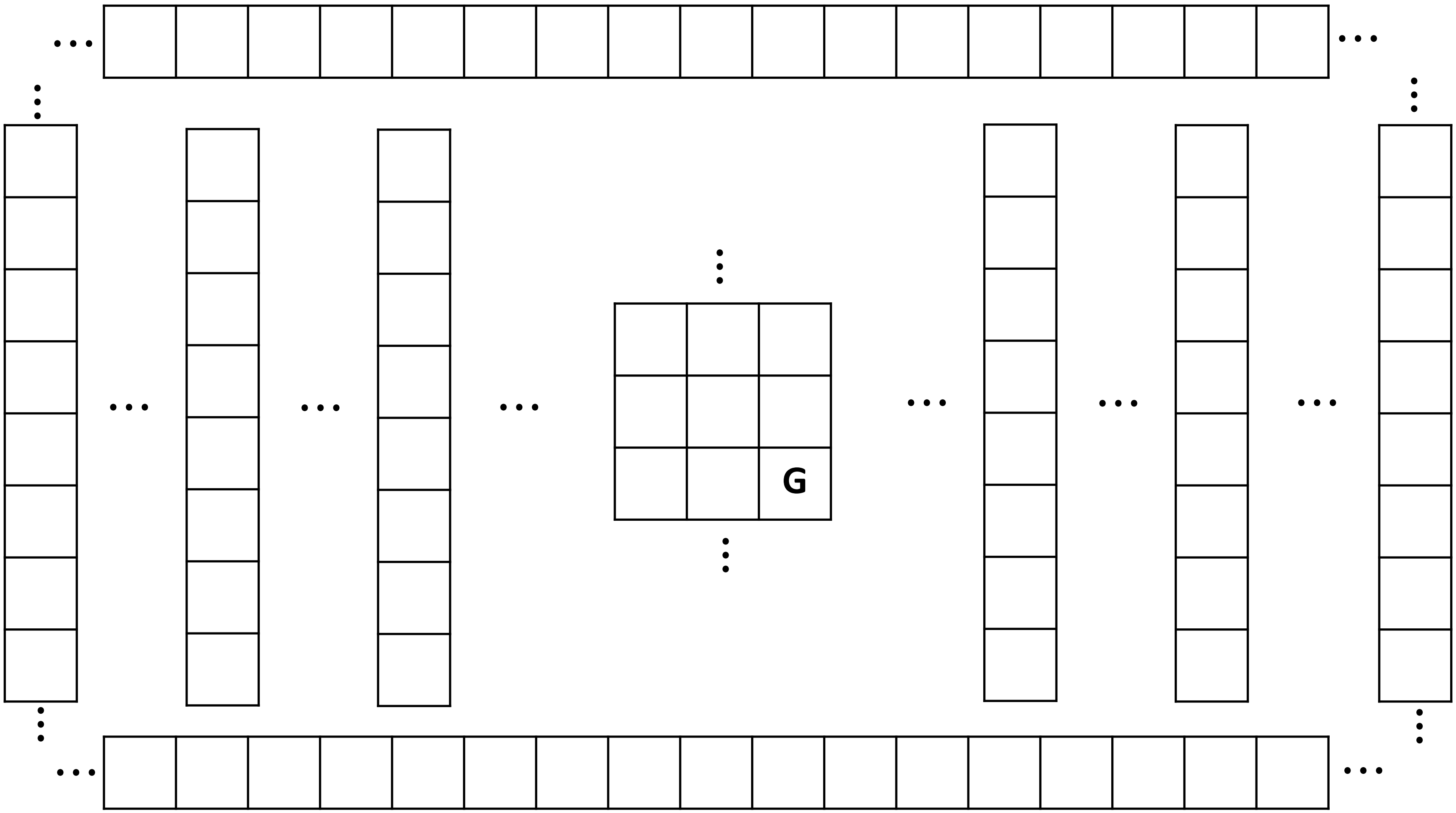
VI and goal regression



Dynamic programming works backwards from the goals.

$$V^*(s_k) = \max_{a_k} R(s_k, a_k) + \gamma \sum_{s_{k+1}} V^*(s_{k+1})\, T(s_k, a_k, s_{k+1})$$

How does VI scale?



How does VI scale?

VI allows one to compute the optimal policy

from every state reaching the goal.

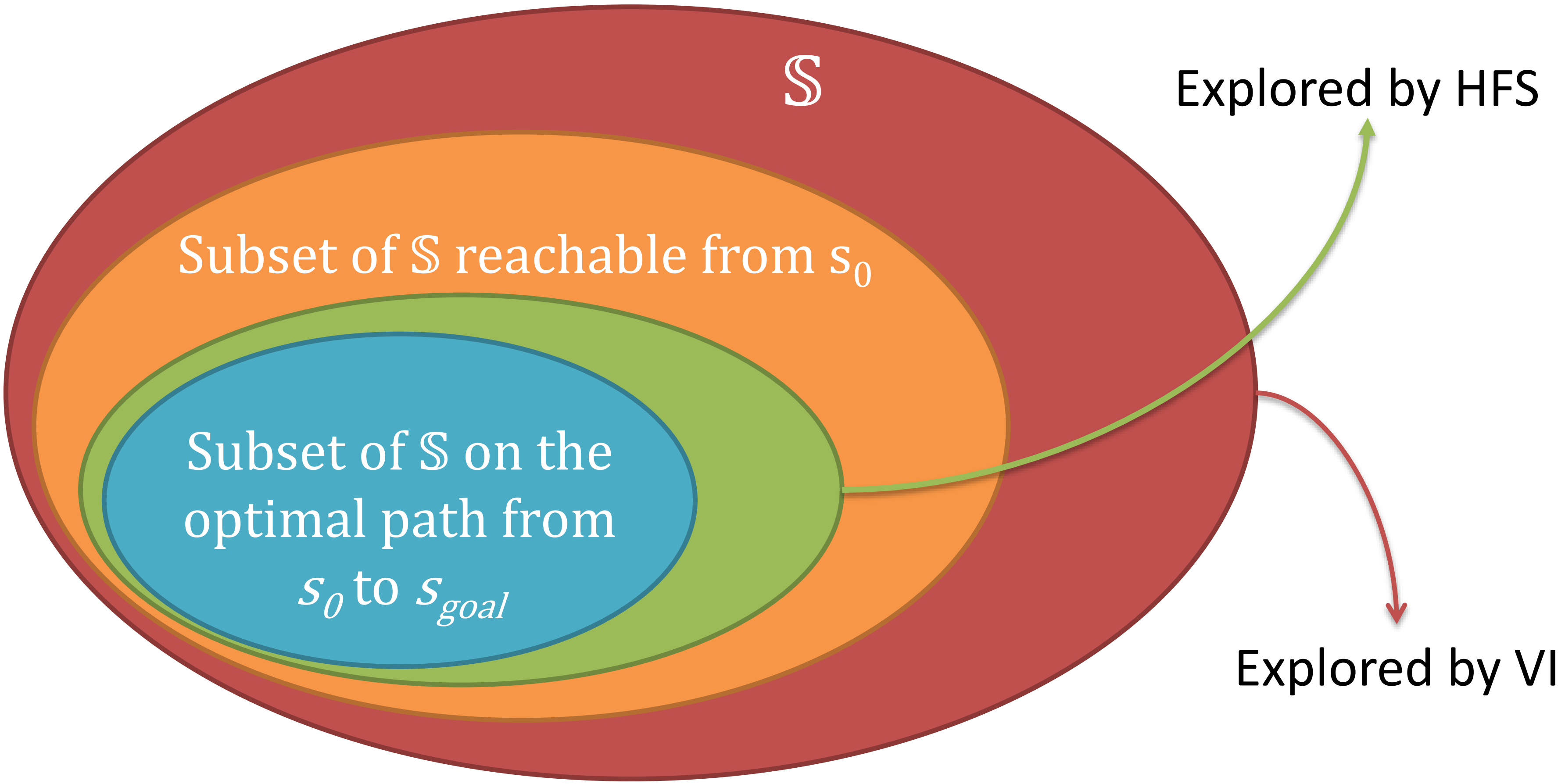
Grows linearly with $|\mathbb{S}|$.

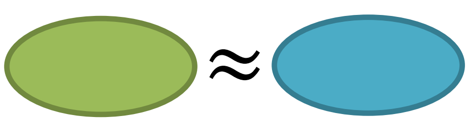
$|\mathbb{S}|$ grows exponentially with the dimensions of $\mathbb{S}$.

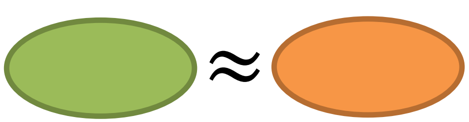
What if we only care about policies starting at one (or a few) possible initial states s_0?

Heuristic forward search!

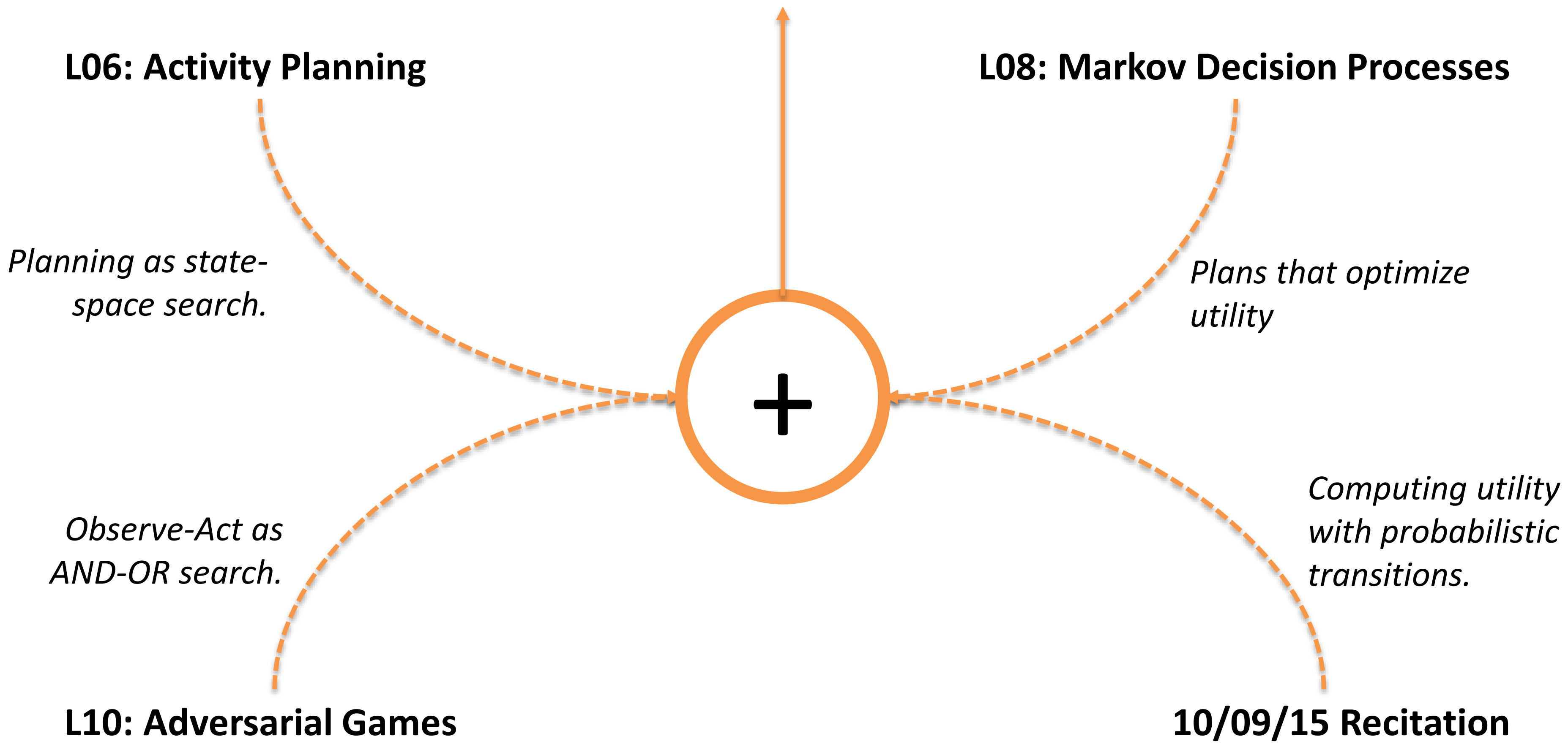
Searching from an initial state s_0



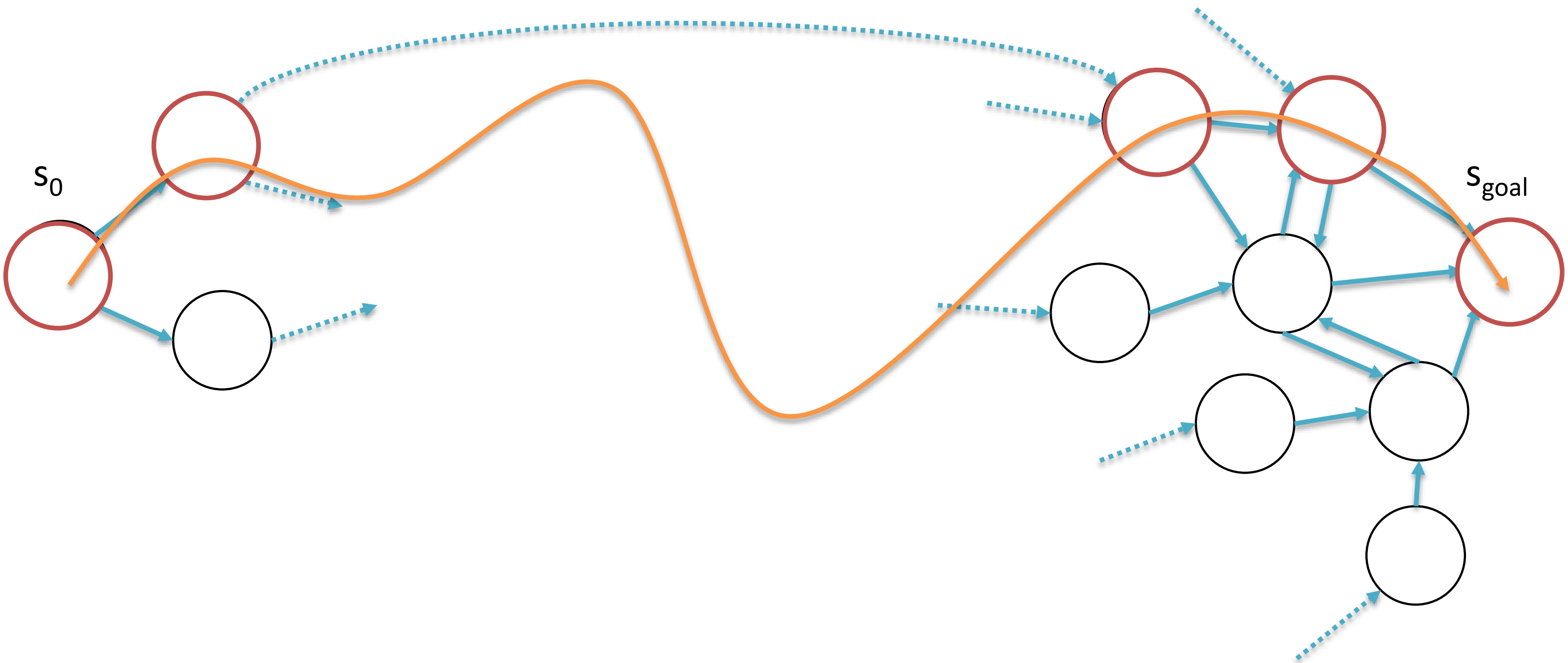
Good heuristic: 

Bad heuristic: 

3. Search-based probabilistic planning

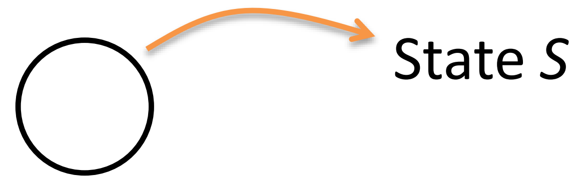


Searching on the state graph



Elements of probabilistic search

$\mathbb{S}$: discrete set of states.



$\mathbb{A}$: discrete set of actions.

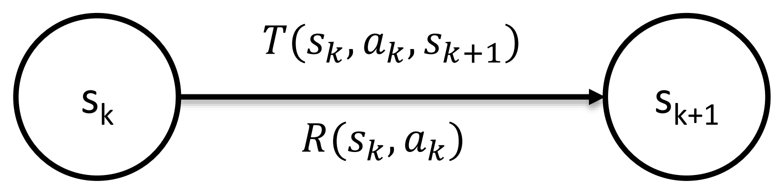
T: $\mathbb{S}\times\mathbb{A}\times\mathbb{S}\rightarrow [0,1]$, *transition function*

Looks familiar?

R: $\mathbb{S}\times\mathbb{A}\rightarrow\mathbb{R}$, *reward function.*

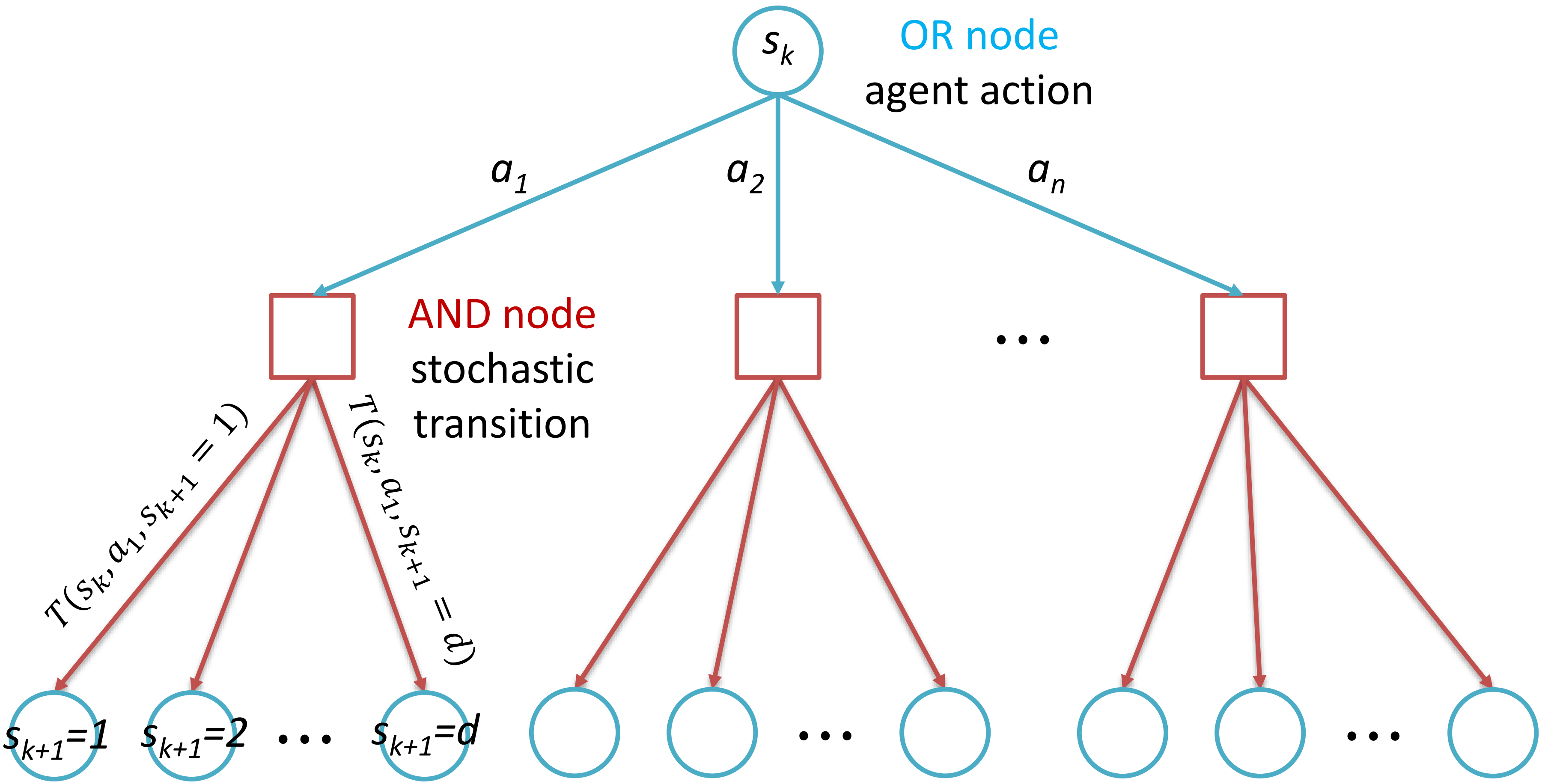
$s_0 \in \mathbb{S}$: initial state.

$\mathbb{S}_g \subseteq \mathbb{S}$: (terminal) goal states.



Could we frame previously-seen shortest path problems like this? How?

(Probabilistic) AND-OR search

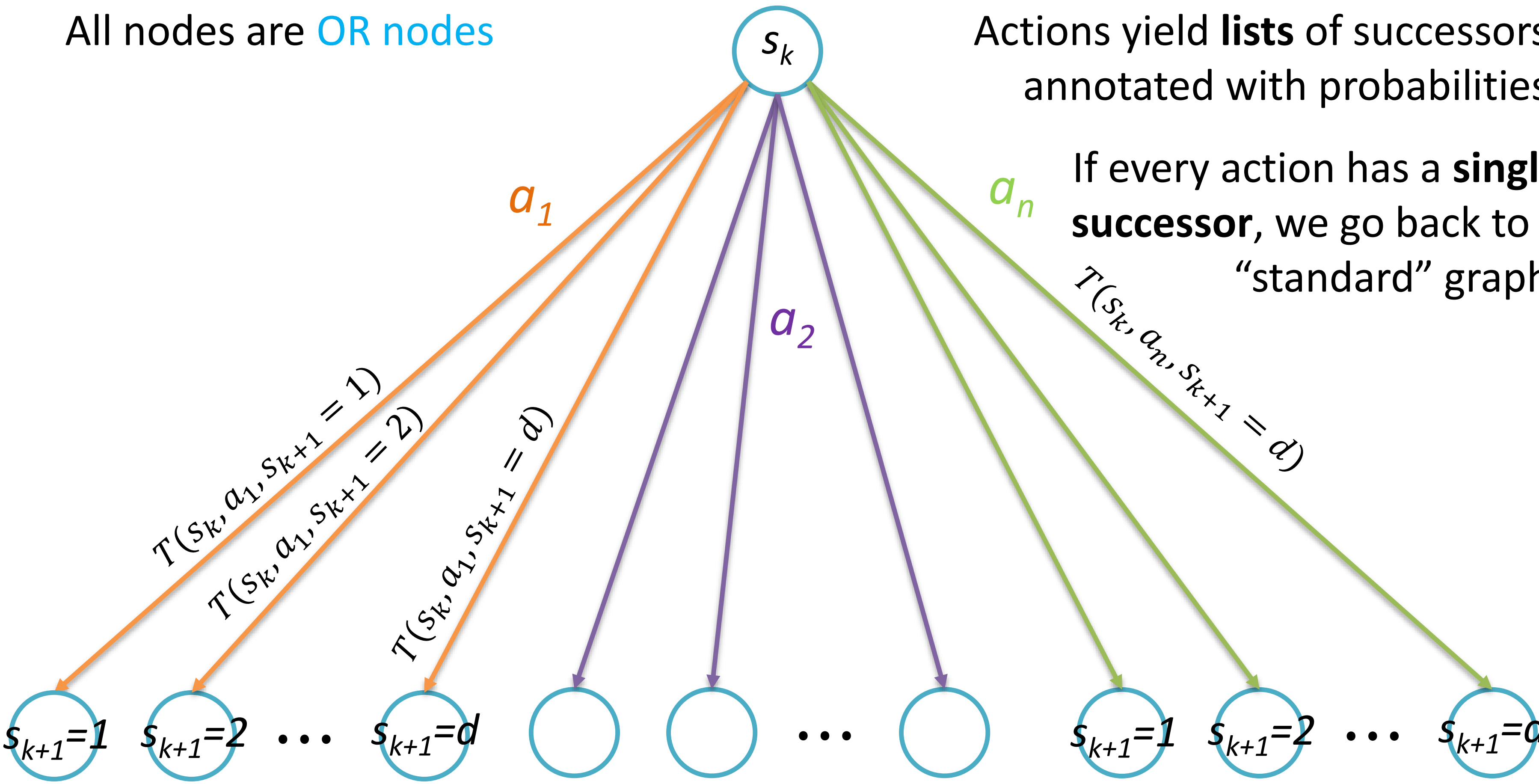


Hypergraph representation

All nodes are OR nodes

Actions yield **lists** of successors annotated with probabilities

If every action has a **single successor**, we go back to a "standard" graph.



4. AO* (Nilson, 1982)

Like A, but for AND-OR search*

AO* in a nutshell

- *Input*: implicit AND-OR search problem

$$< \mathbb{S}, \mathbb{A}, T, R, s_0, \mathbb{S}_g >$$

and an admissible heuristic function $h: \mathbb{S} \rightarrow \mathbb{R}$.

- *Output*: **optimal** policy in the form of an **acyclic hypergraph** mapping states to actions.
  - Cyclic policies: use LAO* (which we'll see in a bit).

- *Strategy*: incrementally build solutions forward from s_0, using h to estimate future utilities (just like A*!). The set of explored solutions form the *explicit* hypergraph G, and the subset of G corresponding to the current estimate of the best policy is called the *greedy* hypergraph g.

Admissible utility estimates

$$\underbrace{V(s_k, a_k)}_{\substack{\text{Expected reward} \\ \text{of executing } a_k \\ \text{at } s_k}} = \underbrace{R(s_k, a_k)}_{\substack{\text{Immediate} \\ \text{reward}}} + \underbrace{\sum_{s_{k+1}} \underbrace{V^*(s_{k+1})}_{\substack{\text{Optimal reward} \\ \text{at } s_{k+1}}} T(s_k, a_k, s_{k+1})}_{\text{Expected optimal future reward}}$$

Unknown

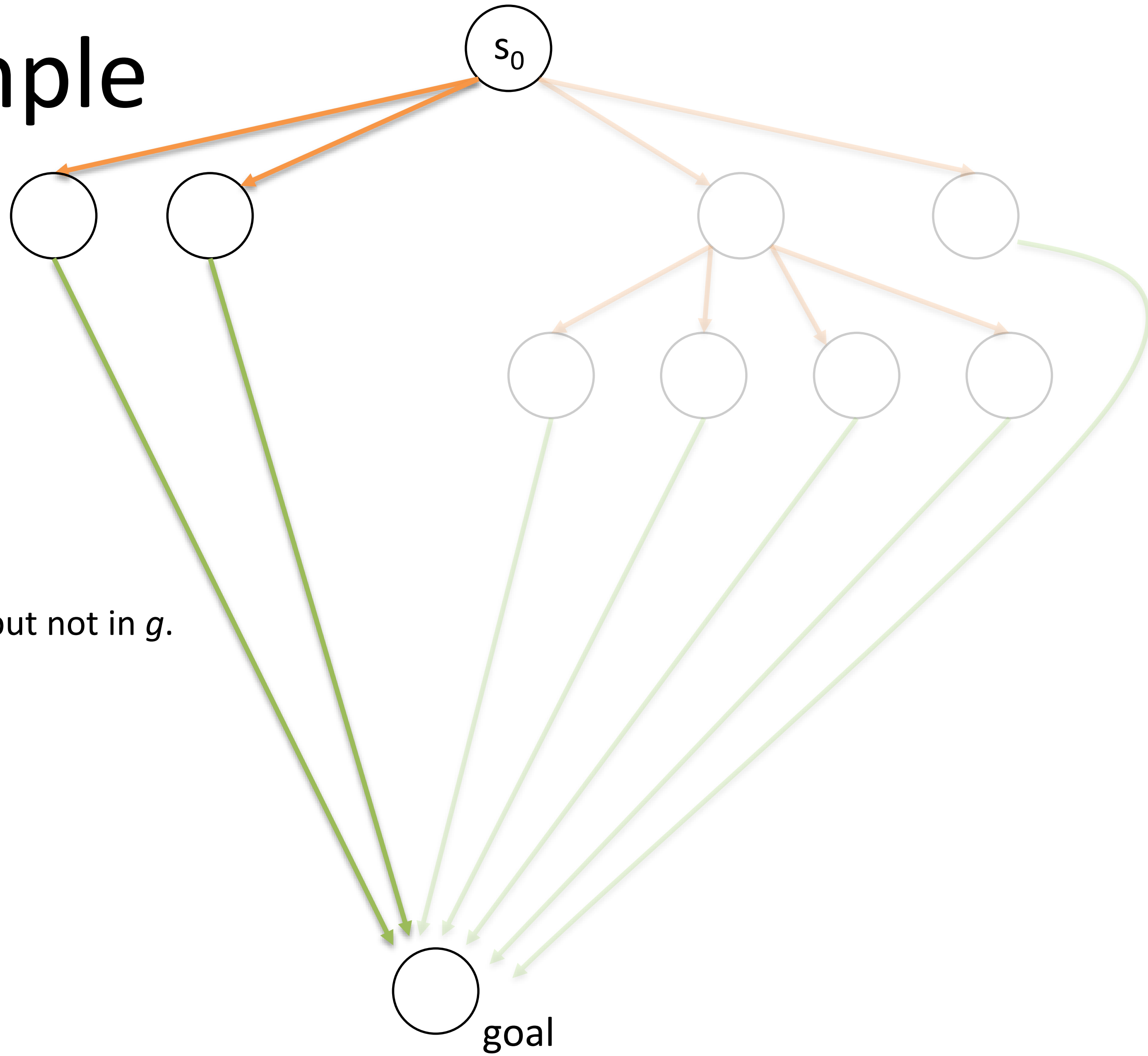
$$h(s_{k+1}) \geq V^*(s_{k+1})$$

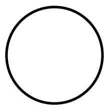
Admissible ("optimistic") estimate of future reward.

Should be "easy" to compute.

$$\boxed{V_h(s_k, a_k) = R(s_k, a_k) + \sum_{s_{k+1}} h(s_{k+1}) T(s_k, a_k, s_{k+1}) \geq V(s_k, a_k)}$$

AO* example





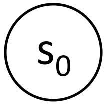
Node in *g*, the greedy graph.

Node in *G*, the explicit graph, but not in *g*.

Known value

Heuristic estimate

Start

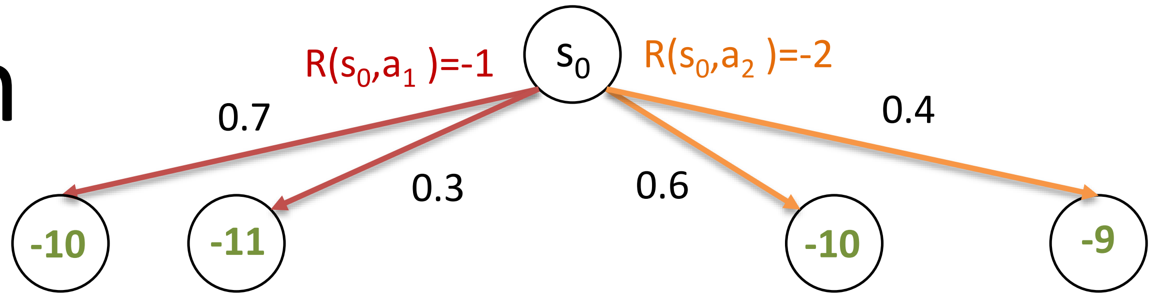


Open nodes: [s_0]

$g=\{s_0: None\}$

G starts just with just the initial state s_0

Expansion



Open nodes: [s_0]

g={s_0: None }

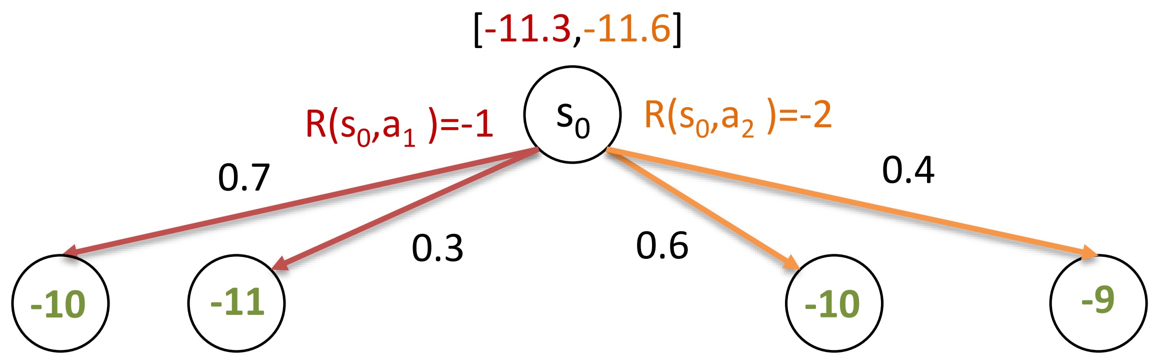
1. *Choose an open node to expand* $\rightarrow$ *s0*

2. *Estimate the value of the leaf nodes using the* ***heuristic h***.

Backup

Open nodes: $[s_0]$

$g=\{s_0\text{: None}\}$



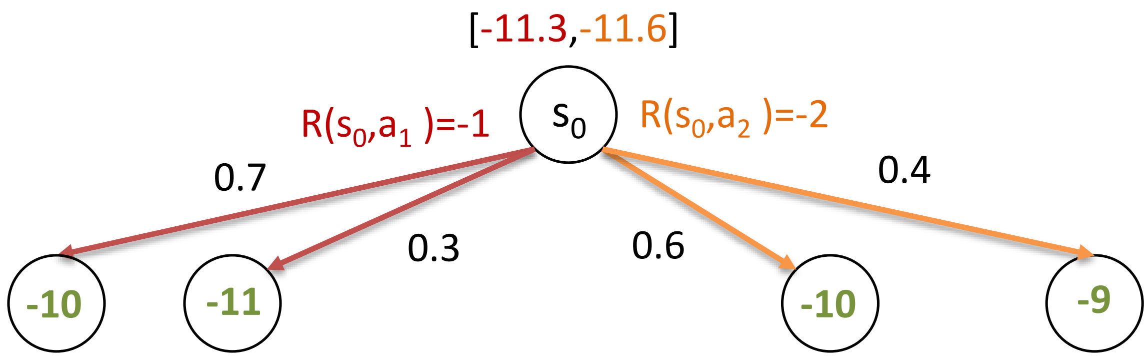
3. Backup values for the currently expanded node (s_0) and all its ancestors that are part of g (no ancestors), recording the best value at each node.

$$V_h(s_k,a_k) = R(s_k,a_k) + \sum_{s_{k+1}} h(s_{k+1})\, T(s_k,a_k,s_{k+1})$$

$$\boxed{V_h(s_0,a_1) = -1 + (-10 * 0.7 - 11 * 0.3) = -11.3}$$

$$V_h(s_0,a_2) = -2 + (-10 * 0.6 - 9 * 0.4) = -11.6$$

Update *g*



Open nodes: $[s_0]$

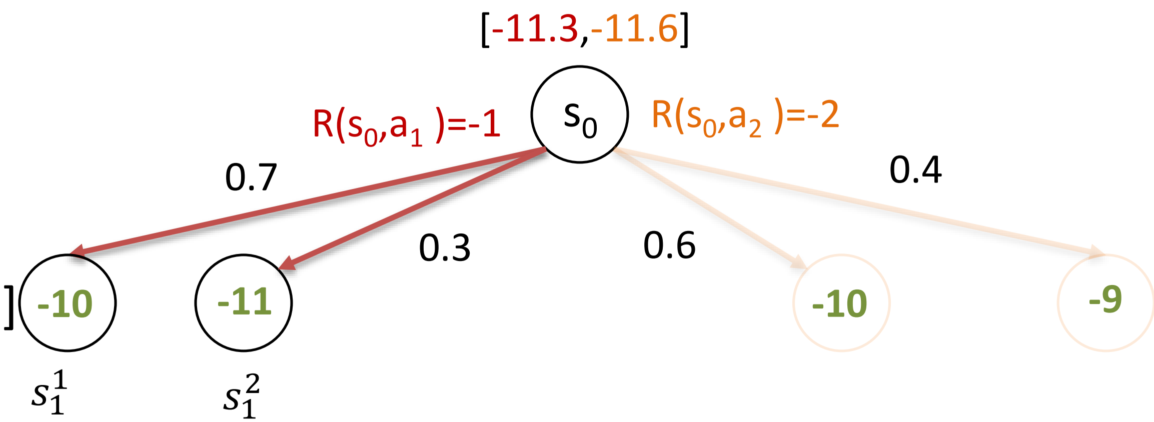
$g=\{s_0: None \}$

4. Update g and the list of open nodes (non-terminal) by selecting the best action at the nodes which got their values updated

$$\boxed{V_h(s_0, a_1) = -1 + (-10 * 0.7 - 11 * 0.3) = -11.3}$$

$$V_h(s_0, a_2) = -2 + (-10 * 0.6 - 9 * 0.4) = -11.6$$

Update *g*



Open nodes: $[s_1^1, s_1^2]$

g={s_0: a1}

4. Update g and the list of open nodes (non-terminal) by selecting the best action at the nodes which got their values updated

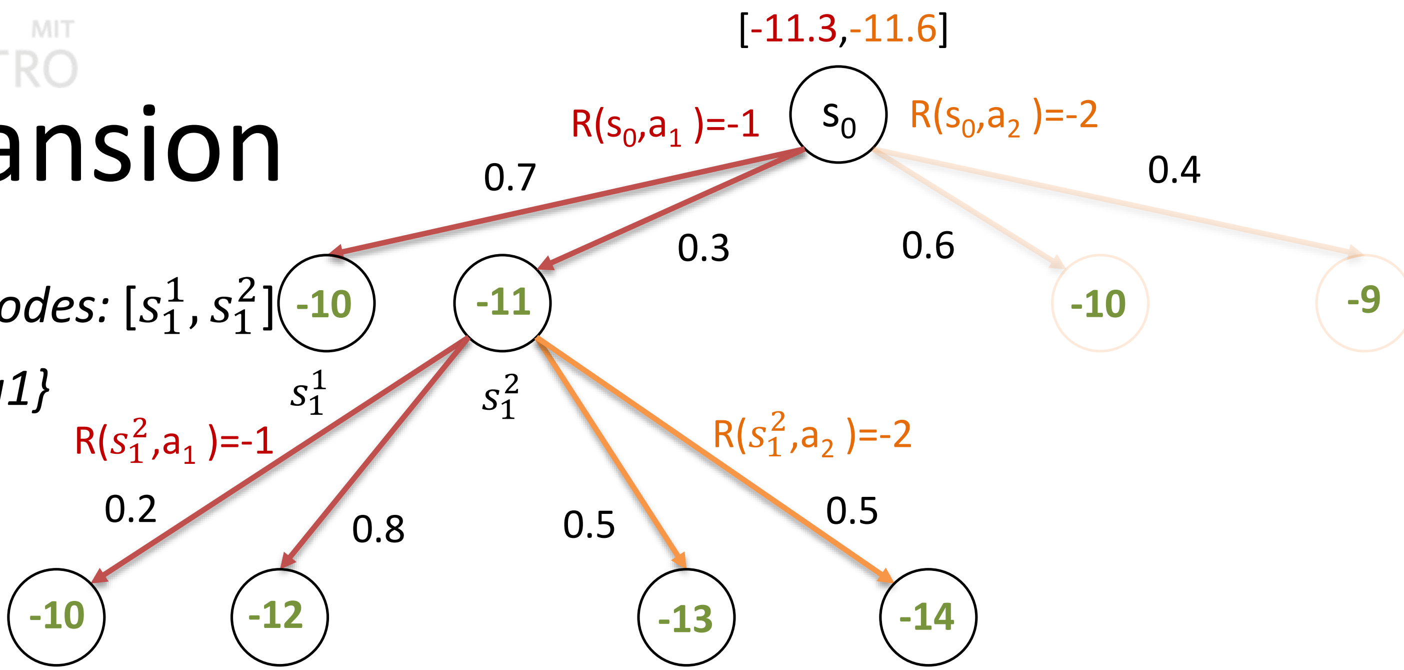
$$\boxed{V_h(s_0, a_1) = -1 + (-10 * 0.7 - 11 * 0.3) = -11.3}$$

$$V_h(s_0, a_2) = -2 + (-10 * 0.6 - 9 * 0.4) = -11.6$$

Expansion

Open nodes: $[s_1^1, s_1^2]$

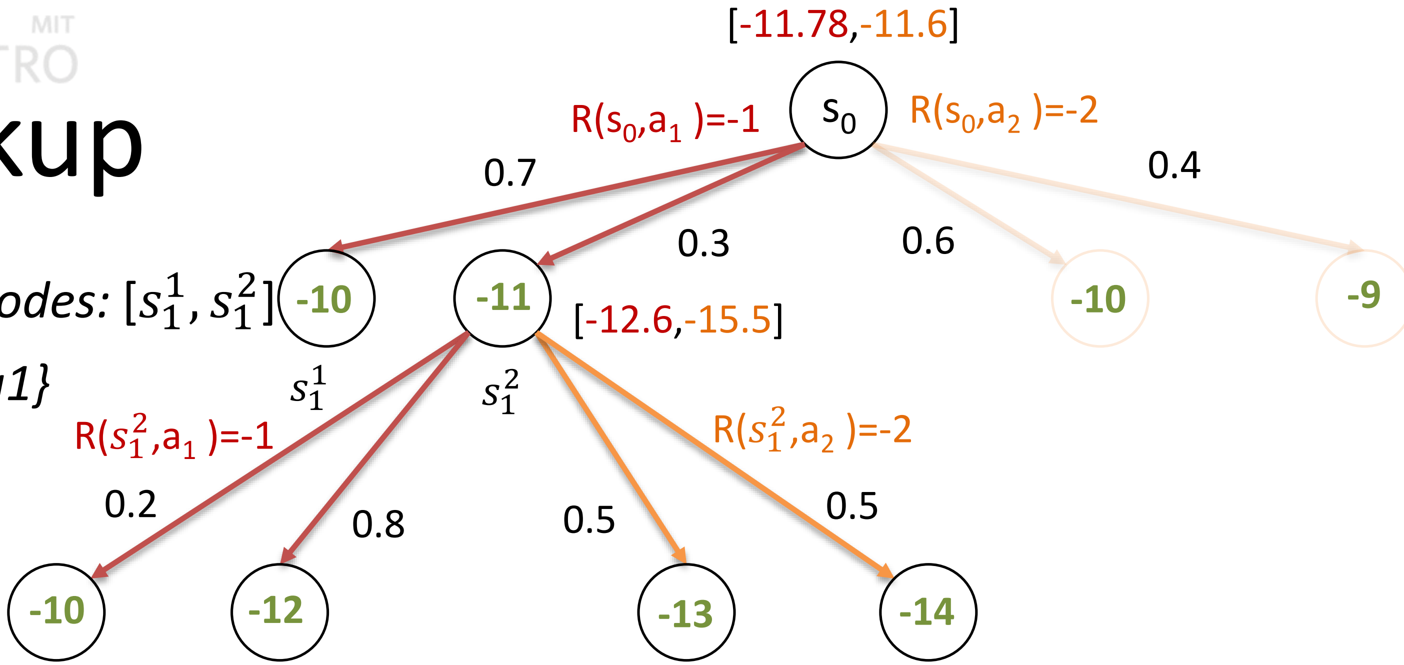
g={s_0: a1}



1. *Choose any open node to expand* $\rightarrow s_1^2$

2. *Estimate the value of the leaf nodes using the* ***heuristic h***.

Backup



Open nodes: $[s_1^1, s_1^2]$

g={s_0: a1}

3. Backup values for the currently expanded node (s_1^2) and all its ancestors that are part of g (s_0), recording the best value at each node.

$$\boxed{V_h(s_1^2, a_1) = -1 + (-10 * 0.2 - 12 * 0.8) = -12.6}$$

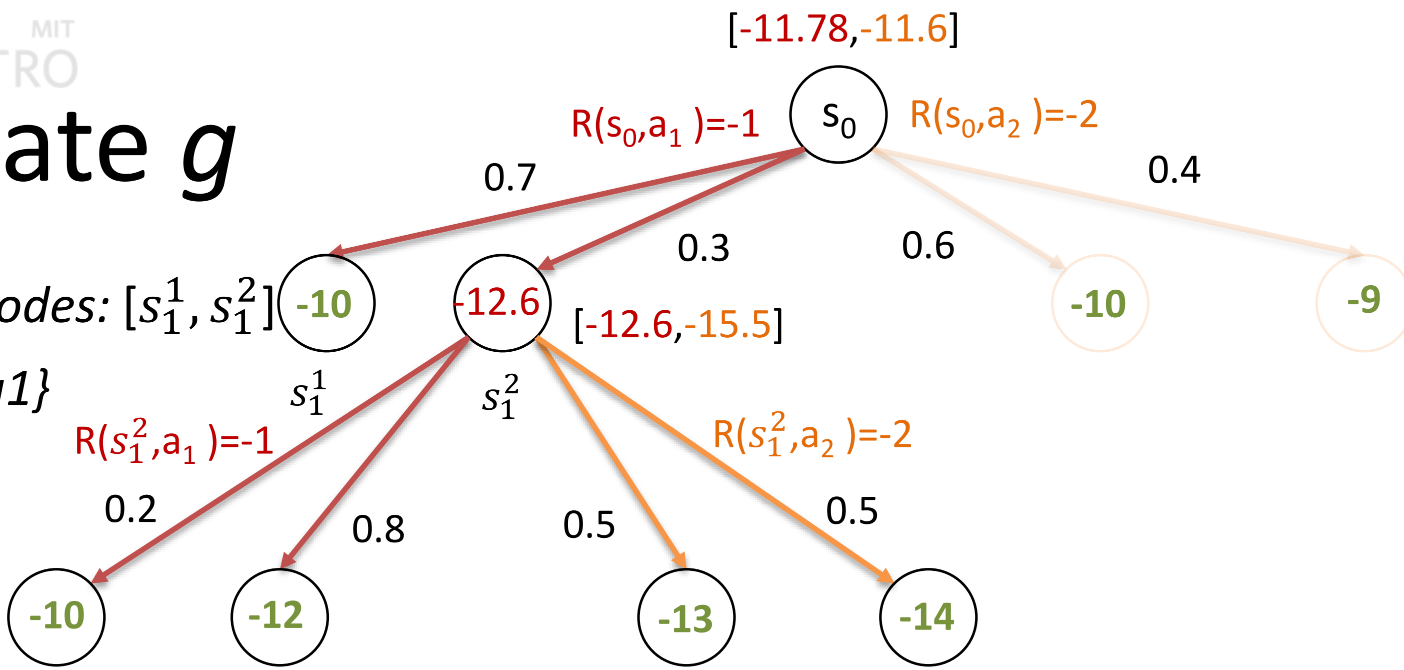
$$V_h(s_1^2, a_2) = -2 + (-13 * 0.5 - 14 * 0.5) = -15.5$$

$$V_h(s_0, a_1) = -1 + (-10 * 0.7 - 12.6 * 0.3) = -11.78$$

$$\boxed{V_h(s_0, a_2) = -2 + (-10 * 0.6 - 9 * 0.4) = -11.6}$$

From leafs to the root

Update *g*



Open nodes: $[s_1^1, s_1^2]$

g={s_0: a1}

4. Update g and the list of open nodes (non-terminal) by selecting the best action at the nodes which got their values updated.

$$\boxed{V_h(s_1^2, a_1) = -1 + (-10 * 0.2 - 12 * 0.8) = -12.6}$$

$$V_h(s_1^2, a_2) = -2 + (-13 * 0.5 - 14 * 0.5) = -15.5$$

$$V_h(s_0, a_1) = -1 + (-10 * 0.7 - 12.6 * 0.3) = -11.78$$

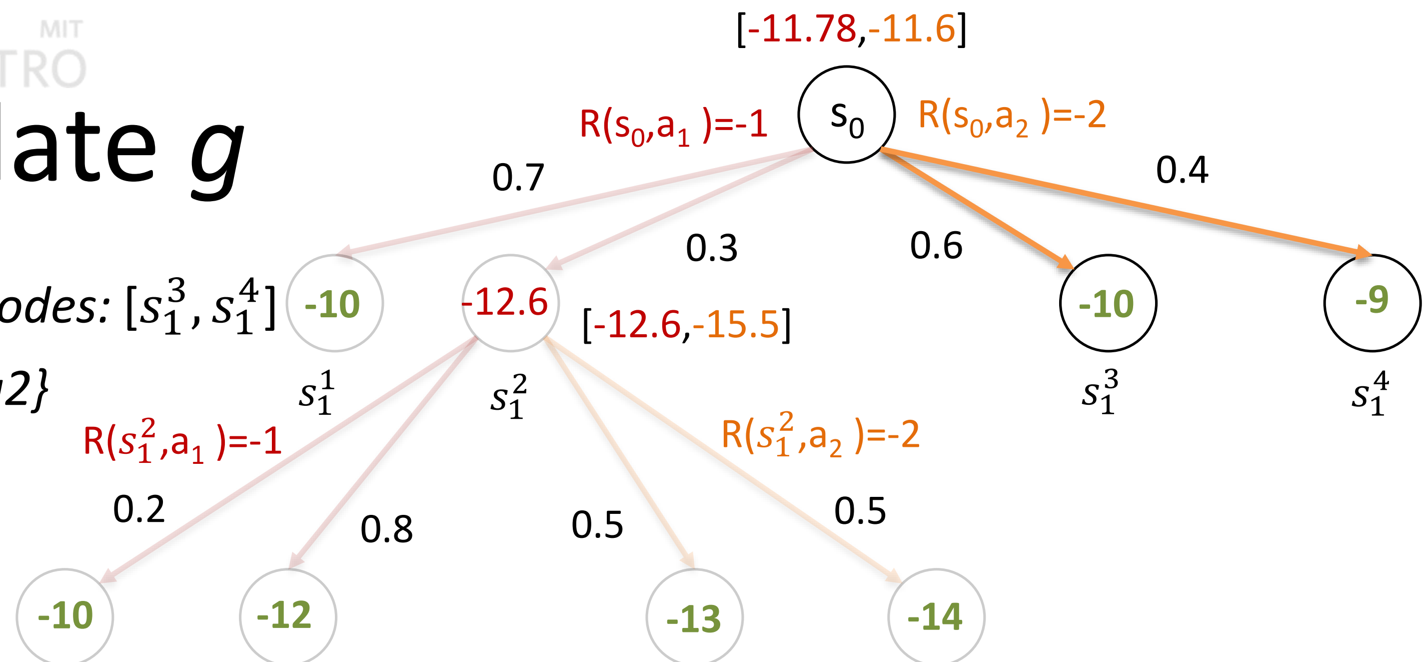
$$\boxed{V_h(s_0, a_2) = -2 + (-10 * 0.6 - 9 * 0.4) = -11.6}$$

From leafs to the root

Update *g*

Open nodes: $[s_1^3, s_1^4]$

g={s_0: a2}



4. Update g and the list of open nodes (non-terminal) by selecting the best action at the nodes which got their values updated.

$$\boxed{V_h(s_1^2, a_1) = -1 + (-10 * 0.2 - 12 * 0.8) = -12.6}$$

$$V_h(s_1^2, a_2) = -2 + (-13 * 0.5 - 14 * 0.5) = -15.5$$

$$V_h(s_0, a_1) = -1 + (-10 * 0.7 - 12.6 * 0.3) = -11.78$$

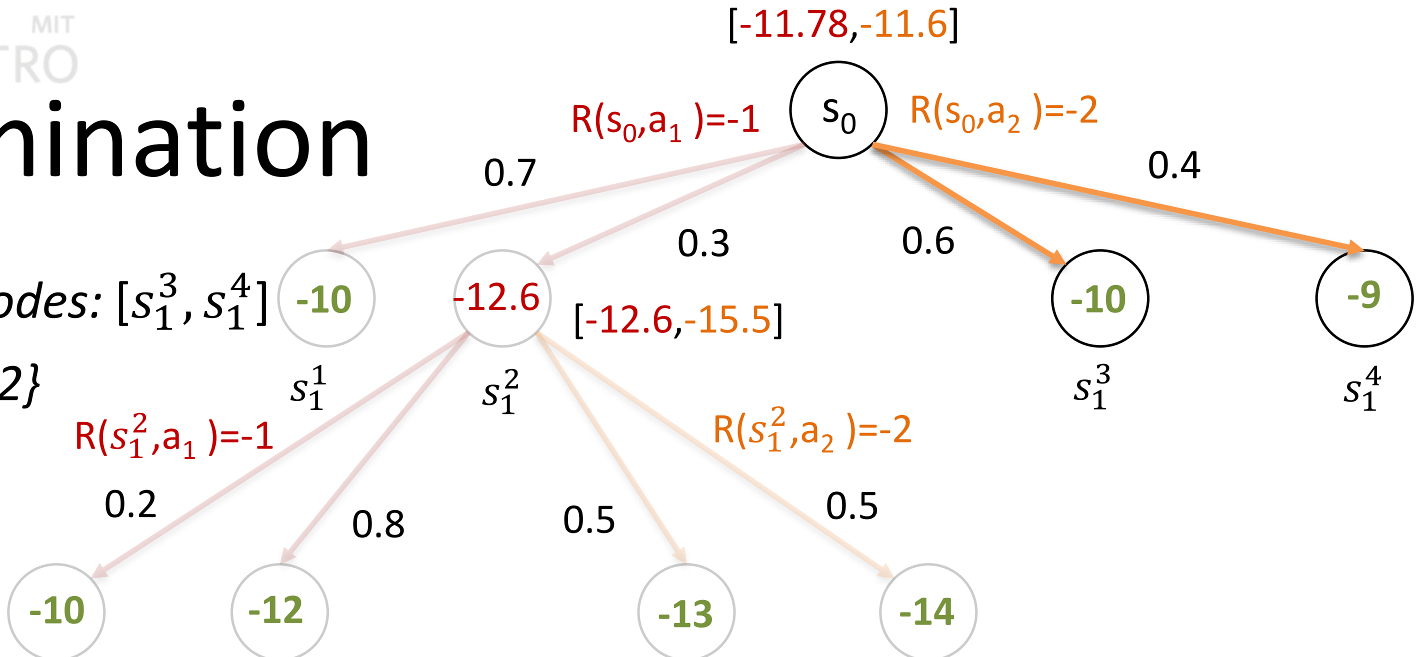
$$\boxed{V_h(s_0, a_2) = -2 + (-10 * 0.6 - 9 * 0.4) = -11.6}$$

From leafs to the root

Termination

Open nodes: $[s_1^3, s_1^4]$

g={s_0: a2}



Search terminates when the list of open nodes is empty, i.e., all leaf nodes in *g* are terminal (goals).

At this point, return *g* as the **optimal** policy π.

AO*'s pseudocode

Input: $< \mathbb{S}, \mathbb{A}, T, R, s_0, \mathbb{S}_g >$, heuristic *h*.

Output: Policy $\pi: \mathbb{S} \rightarrow \mathbb{A}$

Explicit graph $G \leftarrow s_0$, $g \leftarrow$ Best partial policy of G

while best partial policy graph *g* has nonterminal leafs

  $m \leftarrow$ Expand any nonterminal leaf from *g* and add children to *G*

  $Z \leftarrow$ set containing m and all of its predecessors that are part of *g*

  while *Z* is not empty

    $n \leftarrow$ Remove from *Z* a node with no descendants in *Z*

    Update utilities (*V values*) for *n*

    $\pi \leftarrow$ Choose next best action at *n*

  Update *g* with the new π

Heuristics for value-to-go

Bellman backups

g is the graph obtained by following π from s_0

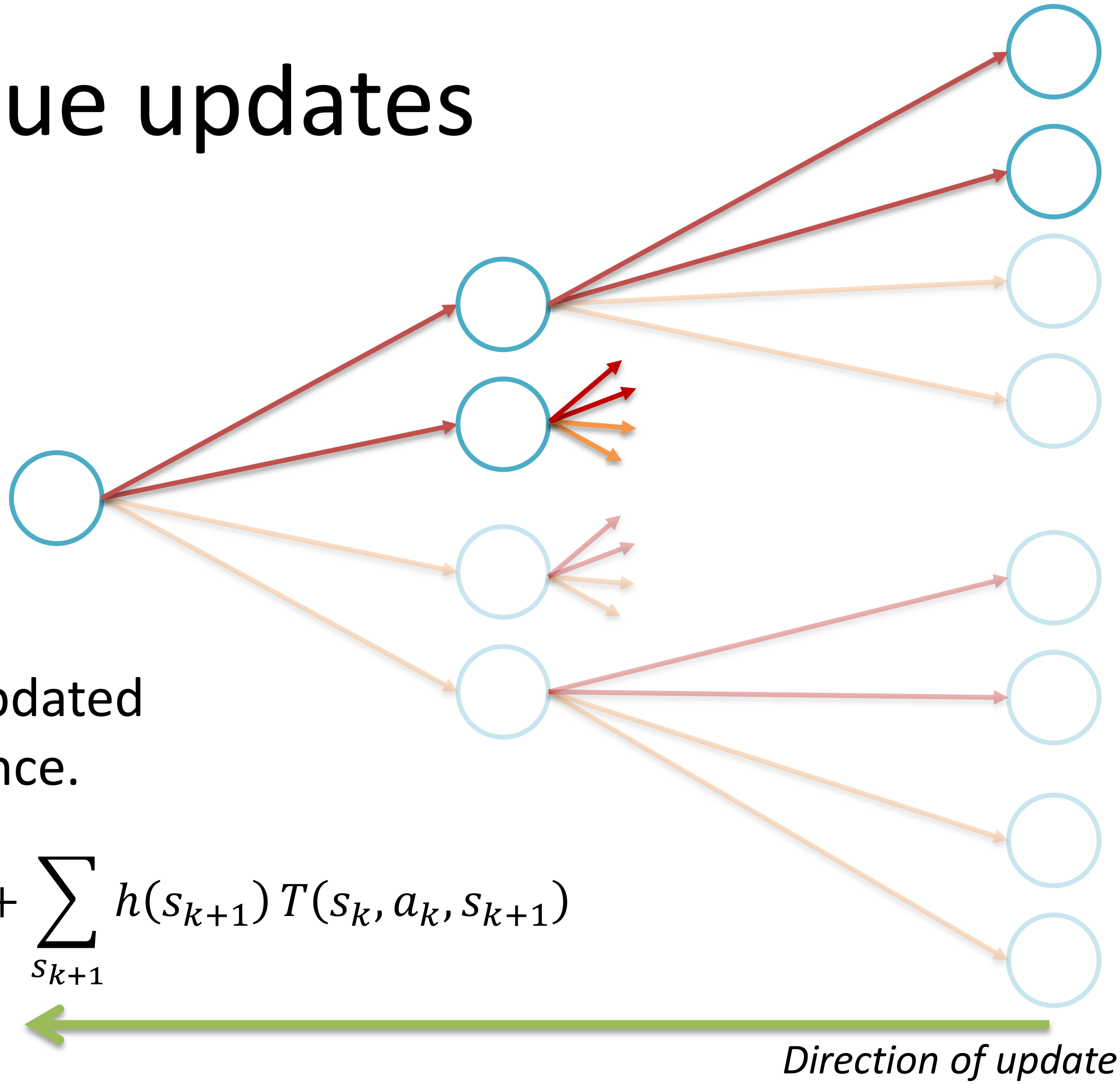
5. LAO* (Hansen & Zilberstein, 2001)

What happens if we find loops in the policy?

Acyclic value updates

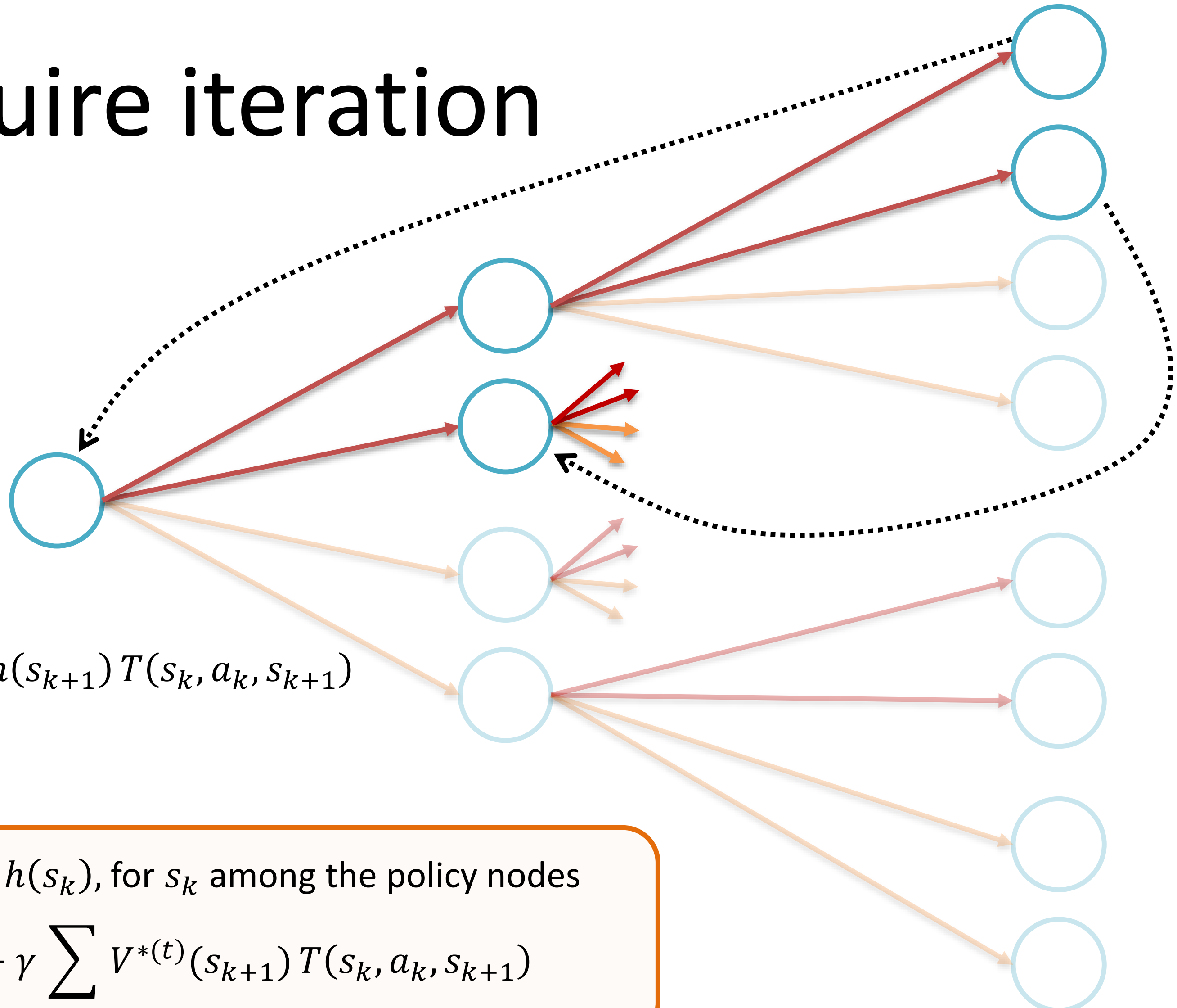
No loops

Policy nodes are updated no more than once.



$$V_h(s_k, a_k) = R(s_k, a_k) + \sum_{s_{k+1}} h(s_{k+1})\, T(s_k, a_k, s_{k+1})$$

Loops require iteration



$$V_h(s_k, a_k) = R(s_k, a_k) + \gamma \sum_{s_{k+1}} h(s_{k+1})\, T(s_k, a_k, s_{k+1})$$

Value iteration

$V^{*(0)}(s_k) = h(s_k)$, for s_k among the policy nodes

$$V^{*(t+1)}(s_k) = \max_{a_k} R(s_k, a_k) + \gamma \sum_{s_{k+1}} V^{*(t)}(s_{k+1})\, T(s_k, a_k, s_{k+1})$$

Updates go both directions

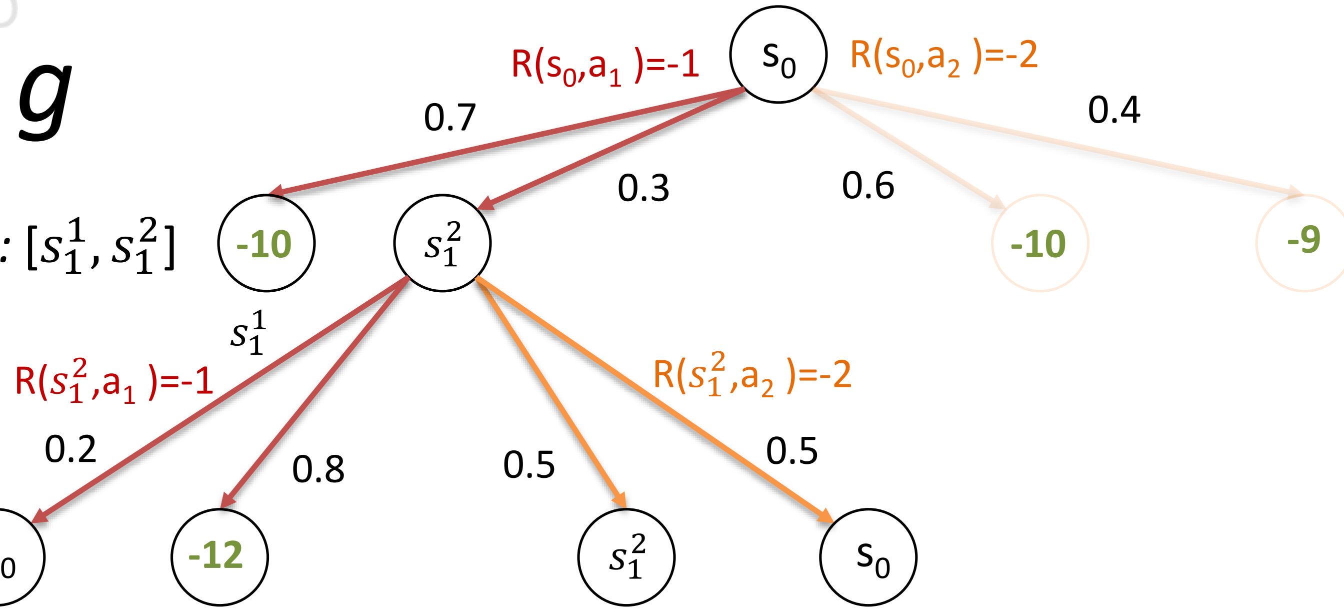
LAO* in a nutshell

- *Input*: MDP $< \mathbb{S}, \mathbb{A}, T, R, \gamma >$, initial state s_0, *and an admissible heuristic* $h: \mathbb{S} \rightarrow \mathbb{R}$.
- *Output*: **optimal** policy mapping states to actions.
- *Strategy*: same as in AO*, but value updates are performed through value or policy iteration.

VI on *g*

Open nodes: $[s_1^1, s_1^2]$

g={s_0: a1}



Perform VI on the expanded node and all of its ancestors in *g*.

Value iteration on policy nodes

$V^{*(0)}(s_k) = h(s_k)$, for s_k among the policy nodes

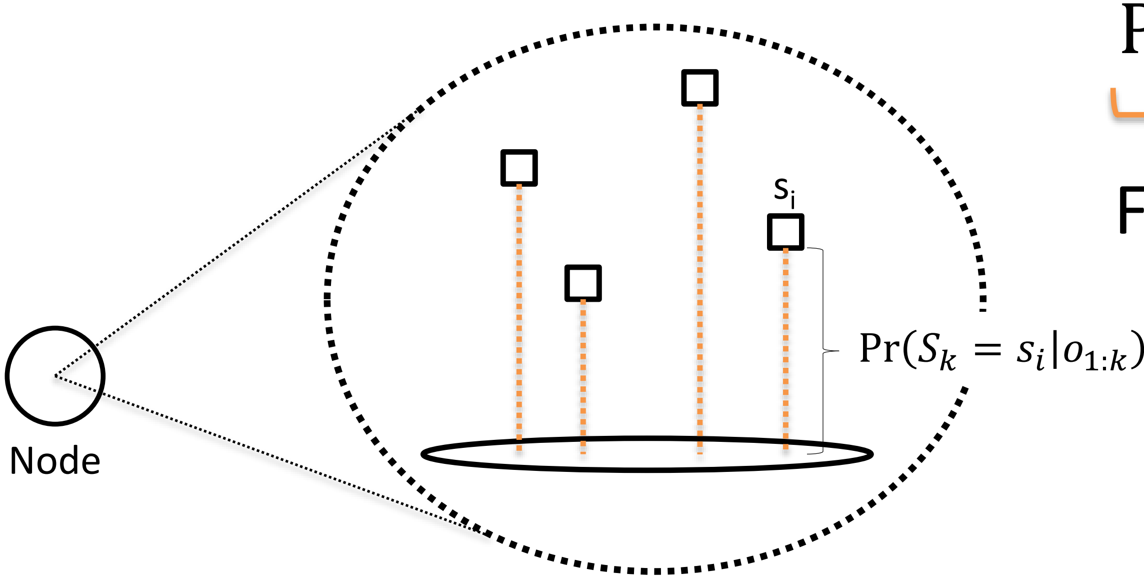
$$V^{*(t+1)}(s_k) = \max_{a_k} R(s_k, a_k) + \gamma \sum_{s_{k+1}} V^{*(t)}(s_{k+1})\, T(s_k, a_k, s_{k+1})$$

6. Extension to hidden state

What changes if the state isn't directly observable?

L09: Hidden Markov Models
Pedro Santana

From states to belief states

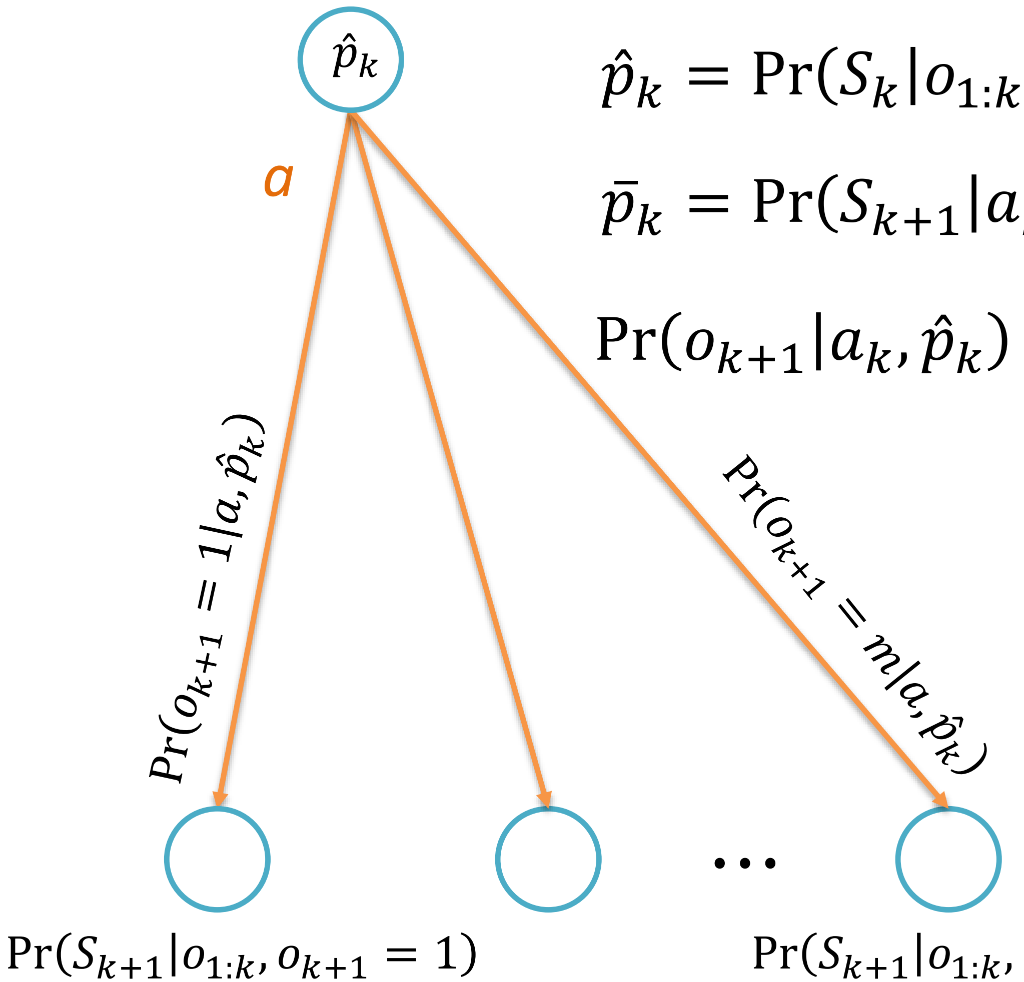


Discrete belief state

$$\underbrace{\Pr(S_k | o_{1:k}) = \hat{p}_k}_{\text{Filtering (forward)}}$$

(see L09: HMMs)

Incorporating HMM observations



$$\hat{p}_k = \Pr(S_k | o_{1:k})$$

$$\bar{p}_k = \Pr(S_{k+1} | a_k, o_{1:k}) = \overbrace{T}^{\Pr(S_{k+1} | S_k, a_k)}(a_k)\hat{p}_k \qquad \text{(prediction)}$$

$$\Pr(o_{k+1} | a_k, \hat{p}_k) = \sum_{s_{k+1}} \underbrace{\Pr(o_{k+1} | s_{k+1})}_{\text{HMM obs. model}} \bar{p}_k(s_{k+1})$$

Estimating belief state utility

Fully observable

$$V_h(s_k, a_k) = R(s_k, a_k) + \sum_{s_{k+1}} T(s_k, a_k, s_{k+1}) h(s_{k+1})$$

$$V_h(\hat{p}_k, a_k) = \sum_{s_k} \hat{p}_k(s_k) R(s_k, a_k) + \sum_{o_{k+1}} \Pr(o_{k+1} | a_k, \hat{p}_k)\, H(\hat{p}_k, o_{k+1})$$

$$H(\hat{p}_k, o_{k+1}) = \sum_{s_{k+1}} \Pr(S_{k+1} = s_{k+1} | o_{1:k}, o_{k+1})\, h(s_{k+1})$$

Partially observable

Partially observable AO* in a nutshell

- *Input*: $<\mathbb{S}, \mathbb{A}, T, R, O, \mathbb{S}_g, h, \hat{p}_0>$, where O is the HMM observation model and $\hat{p}_0$ is the initial belief.

- *Output*: **optimal** policy in the form of an **acyclic hypergraph** mapping beliefs to actions.

- *Strategy*: same as in AO*, replacing: s_0 by $\hat{p}_0$; $T(s_k, a_k, s_{k+1})$ by $\Pr(o_{k+1}|a_k, \hat{p}_k)$; $V_h(s_k, a_k)$ by $V_h(\hat{p}_k, a_k)$.