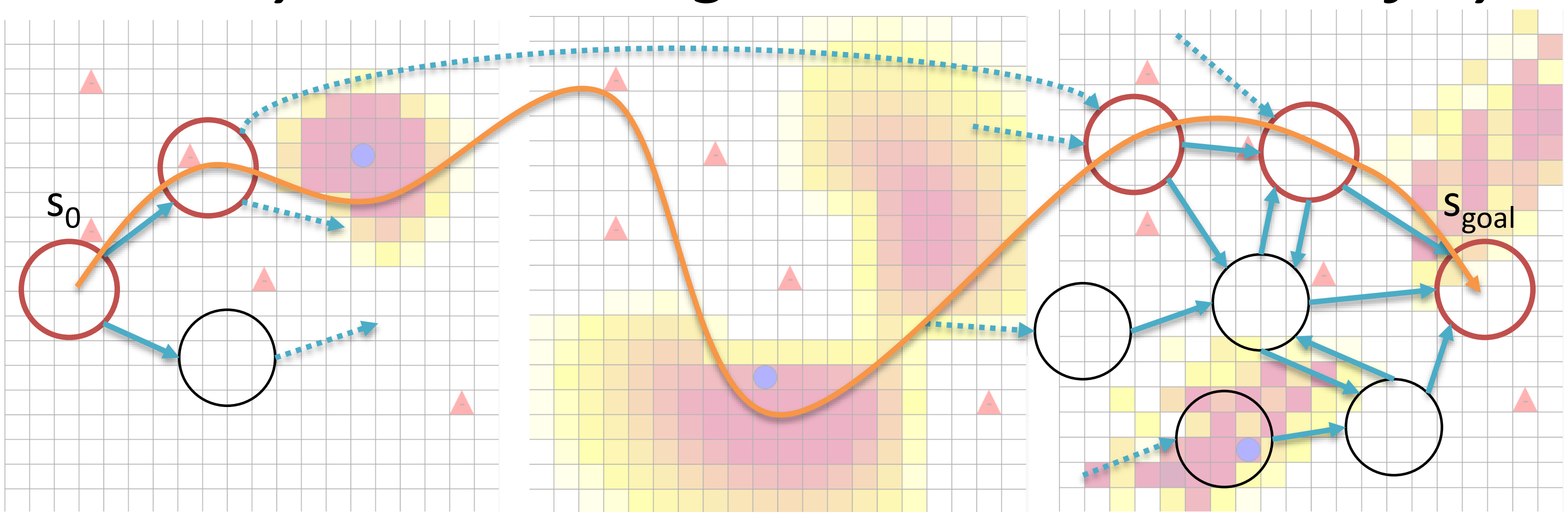
Risk-aware AO* (RAO*)

When you want to get there well and safely



16.412/6.834 Cognitive Robotics

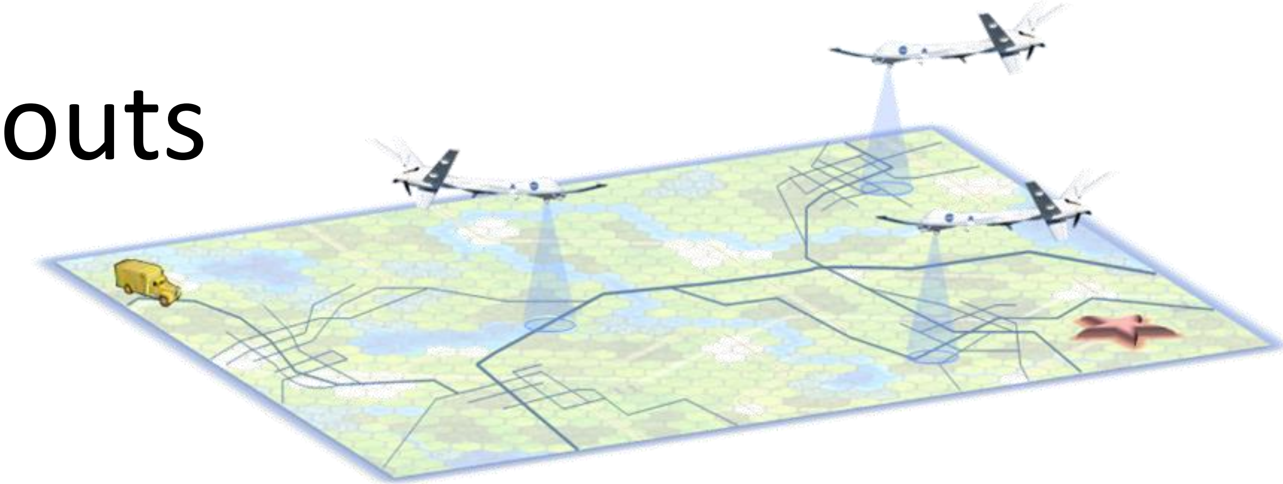
Pedro Santana (psantana@mit.edu)

May the 4th be with you, 2016.

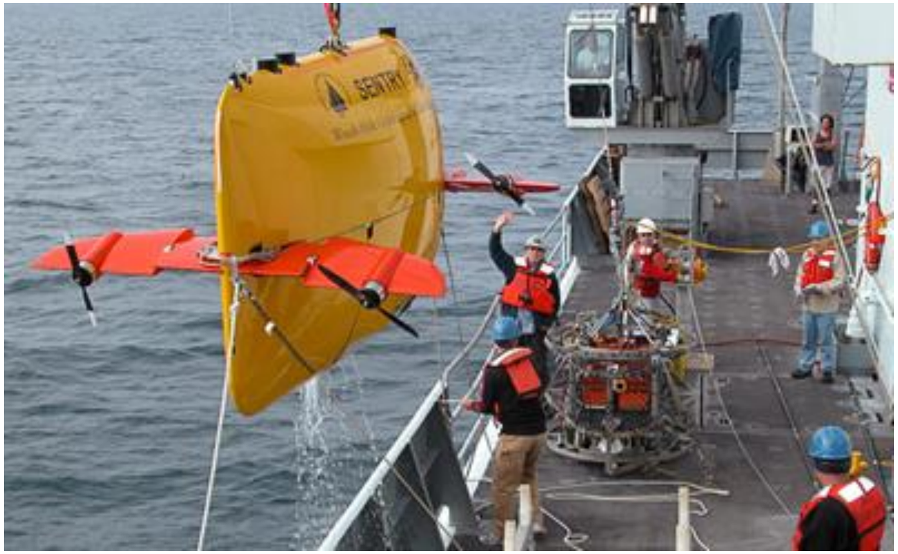
1. Motivation

Where can risk-aware planning be useful?

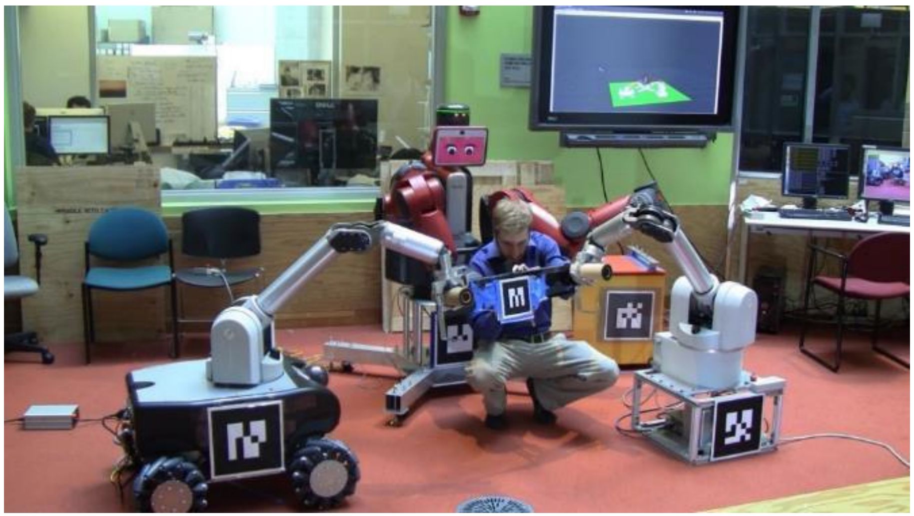
Science scouts

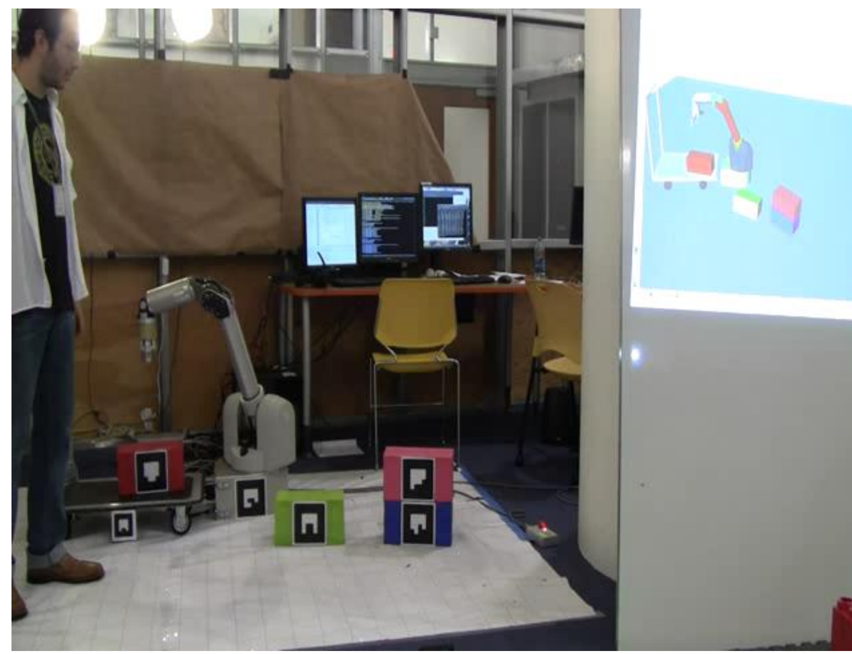


Courtesy of Andrew J. Wang

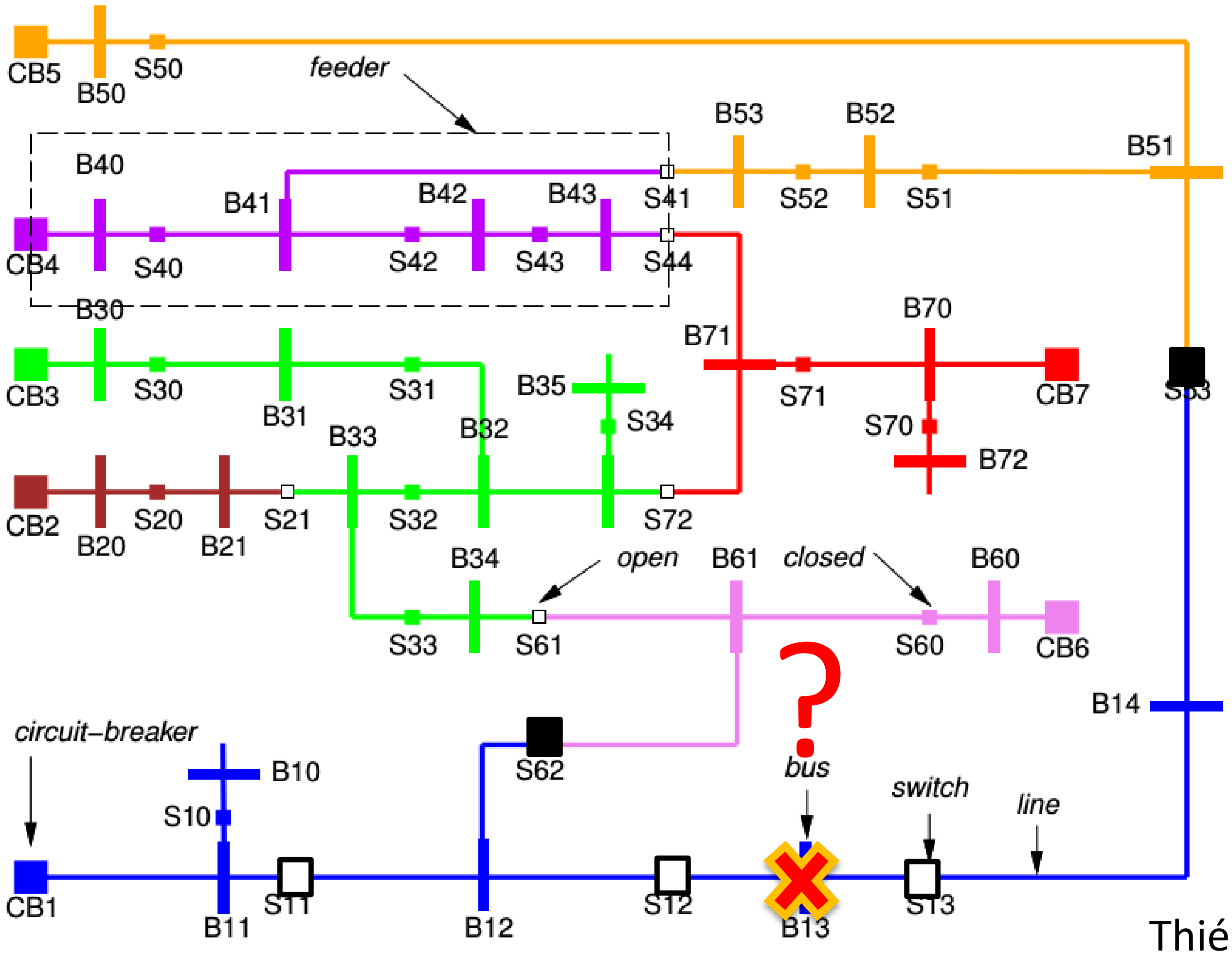


Collaborative manufacturing



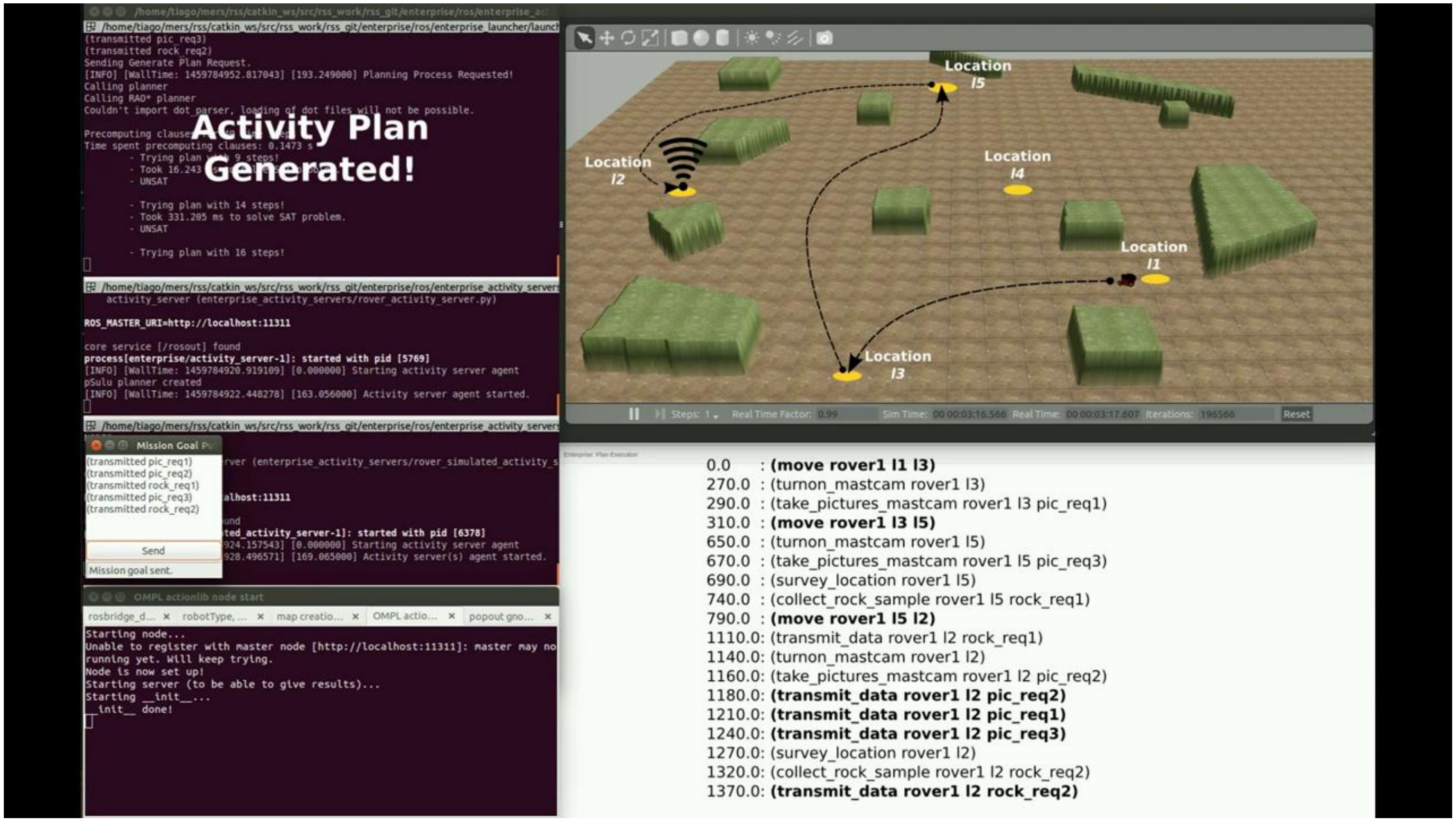


Power supply restoration



Thiébaux & Cordier (2001)

Resilient Space Systems (RSS) demo



Joint work between JPL, Caltech, WHOI, and MIT.

Our goal for today

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

Today’s topics

1. Motivation

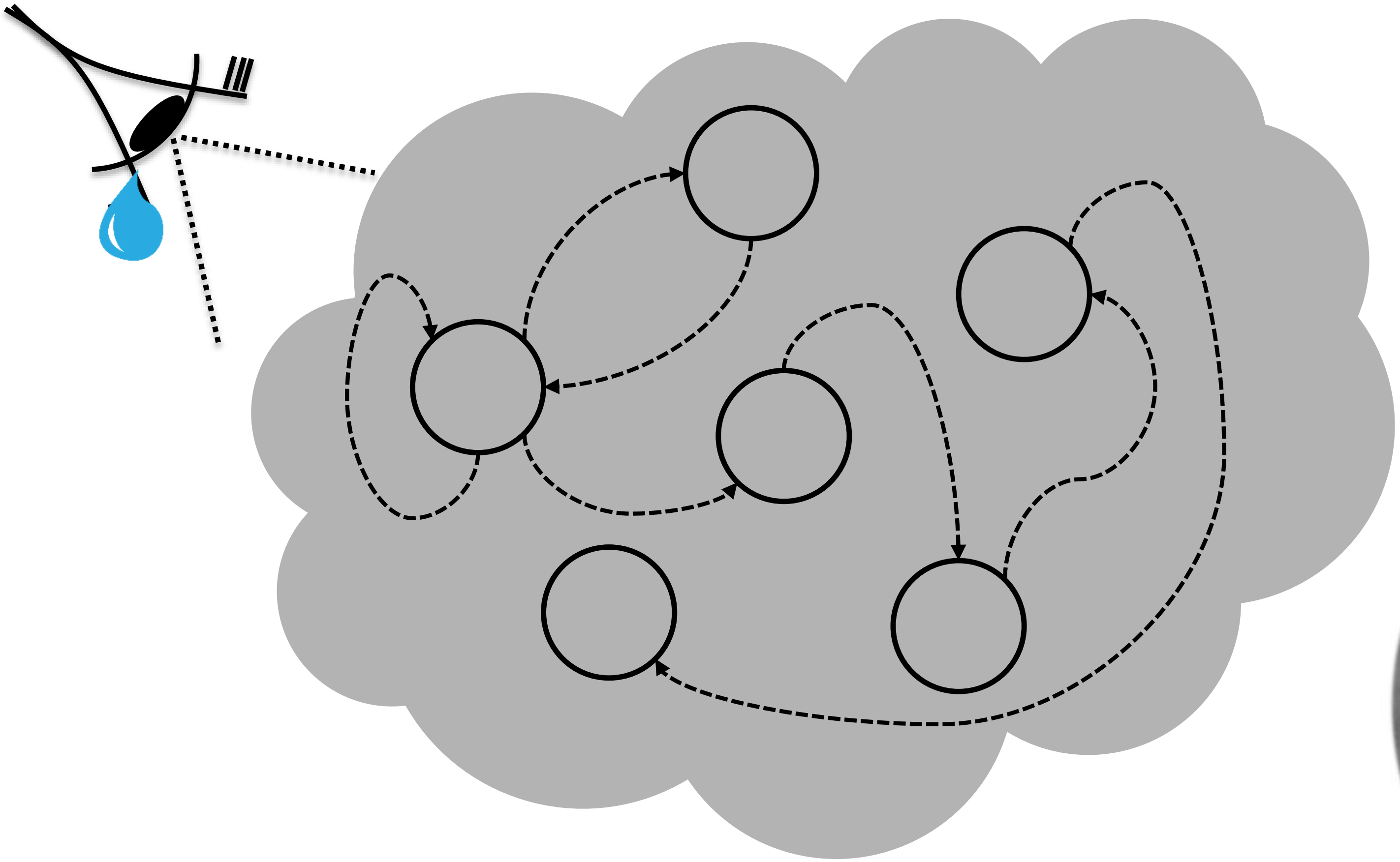
2. Handling belief states

3. RAO*

2. Handling belief states

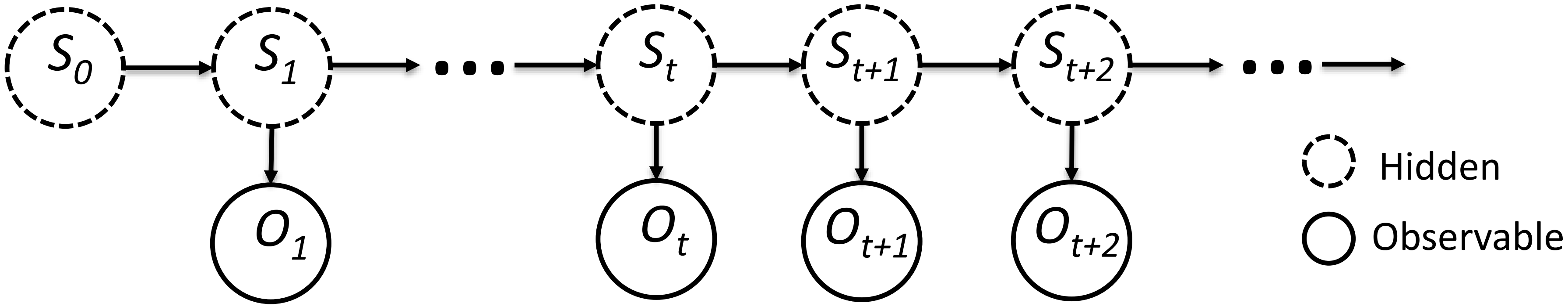
"Probability is common sense reduced to calculation."
—— Pierre-Simon Laplace

Hidden Markov models (HMMs)



Andrey Markov

Observing hidden Markov chains



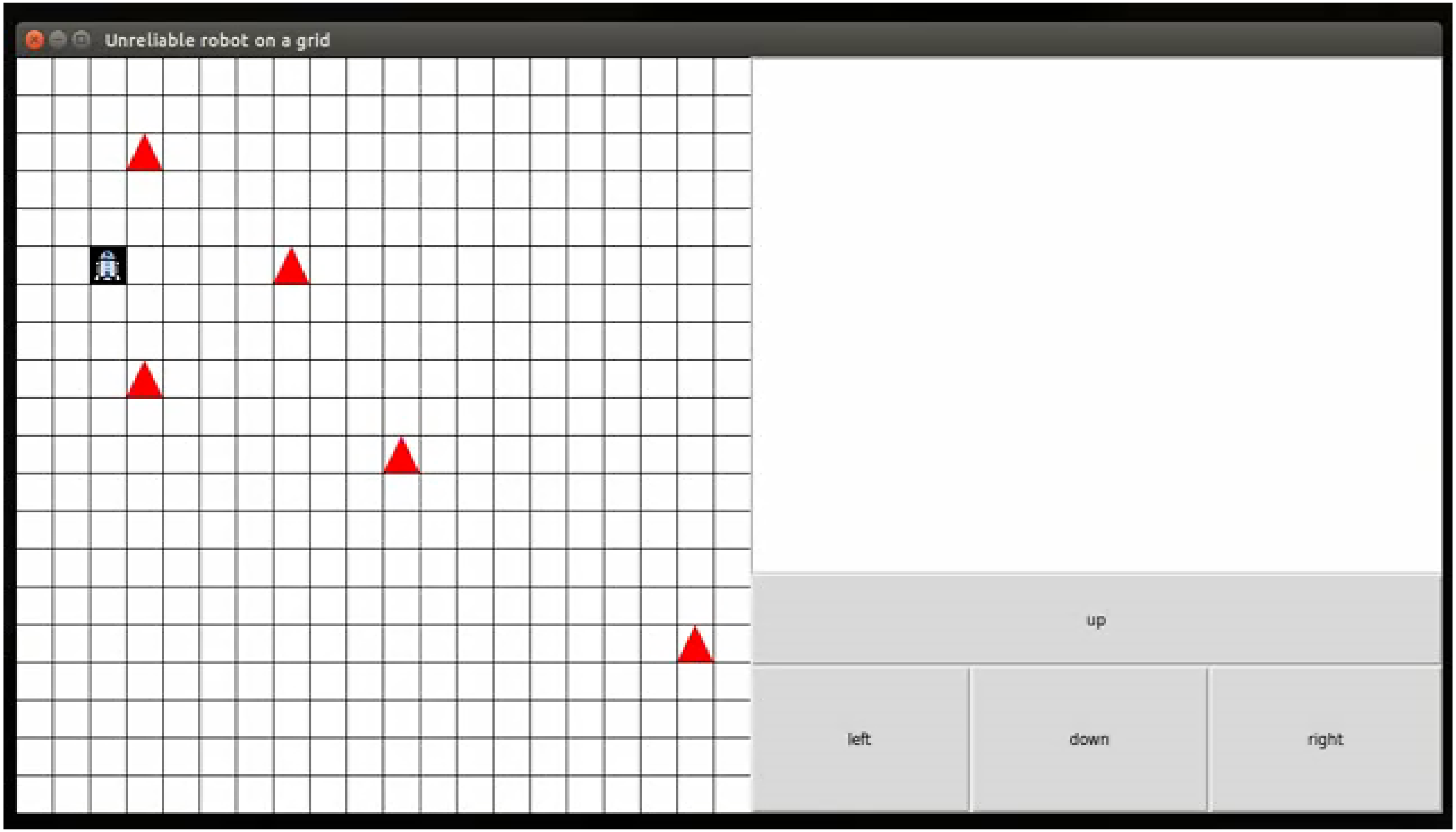
Definition: Hidden Markov Model (HMM)

A sequence of random variables $O_1, O_2, \ldots, O_t, \ldots$, is an HMM if the distribution of O_t is completely defined by the current (hidden) state S_t according to

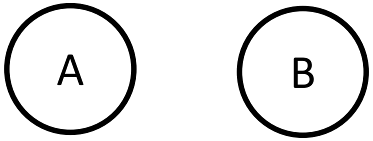
$$\Pr(O_t | S_t),$$

where S_t is part of an underlying Markov chain.

Robot navigation



Bayes' rule



A, B: random variables

$$\overbrace{\Pr(A, B)}^{\text{Joint}} = \overbrace{\Pr(\mathrm{A}|\mathrm{B})}^{\text{Conditional}}\overbrace{\Pr(\mathrm{B})}^{\text{Marginal}}$$

$$\Pr(A, B) = \Pr(\mathrm{B}|\mathrm{A})\Pr(\mathrm{A})$$

$$\Pr(\mathrm{A}|\mathrm{B})\Pr(\mathrm{B}) = \Pr(\mathrm{B}|\mathrm{A})\Pr(\mathrm{A})$$

Bayes' rule!

$$\Pr(\mathrm{A}|\mathrm{B}) = \frac{\Pr(\mathrm{B}|\mathrm{A})\Pr(\mathrm{A})}{\Pr(\mathrm{B})} \propto \Pr(\mathrm{B}|\mathrm{A})\Pr(\mathrm{A})$$

Notation

Random variable!

$\Pr(S_t|\cdot)$ → Probability distribution of S_t

Vector of *d* probability values.

$\underbrace{\Pr(S_t = s|\cdot) = \Pr(\mathrm{s}_\mathrm{t}|\cdot)}_{\text{Probability} \in [0,1]}$ → Probability of observing $S_t = s$ according to $\Pr(S_t|\cdot)$

Filtering (*forward*)

"Given the available history of observations, what's the belief about the current hidden state?"

$$\Pr(S_t|o_{1:t}) = \hat{p}_t$$

$$\begin{aligned}\Pr(S_t|o_{1:t}) &= \Pr(S_t|o_t, o_{1:t-1}) \\ &\propto \Pr(o_t|S_t, o_{1:t-1})\Pr(S_t|o_{1:t-1}) && \textit{Bayes} \\ &= \Pr(o_t|S_t)\Pr(S_t|o_{1:t-1}) && \textit{Obs. model}\end{aligned}$$

$$\begin{aligned}\Pr(S_t|o_{1:t-1}) &= \sum_{i=1}^{d} \Pr(S_t|S_{t-1}=i, o_{1:t-1}) \Pr(S_{t-1}=i|o_{1:t-1}) && \textit{Marg.} \\ &= \sum_{i=1}^{d} \Pr(S_t|S_{t-1}=i) \underbrace{\Pr(S_{t-1}=i|o_{1:t-1})}_{\textit{Recursion!}} && \textit{Trans. model}\end{aligned}$$

Filtering

"Given the available history of observations, what's the belief about the current hidden state?"

$$\Pr(S_t|o_{1:t}) = \hat{p}_t$$

1. One-step prediction:

$$\Pr(S_t|o_{1:t-1}) = \overline{p_t} = \sum_{i=1}^{d} \Pr(S_t|S_{t-1} = i)\Pr(S_{t-1} = i|o_{1:t-1}) = T\hat{p}_{t-1}$$

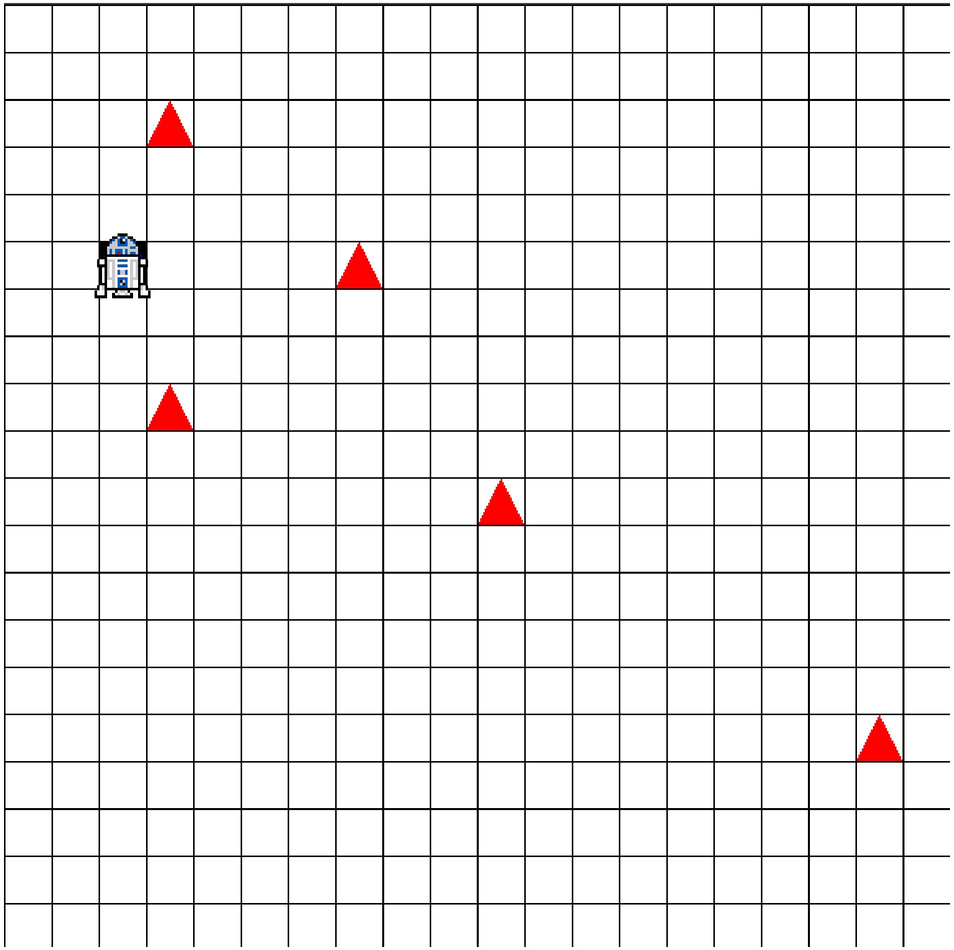
2. Measurement update:

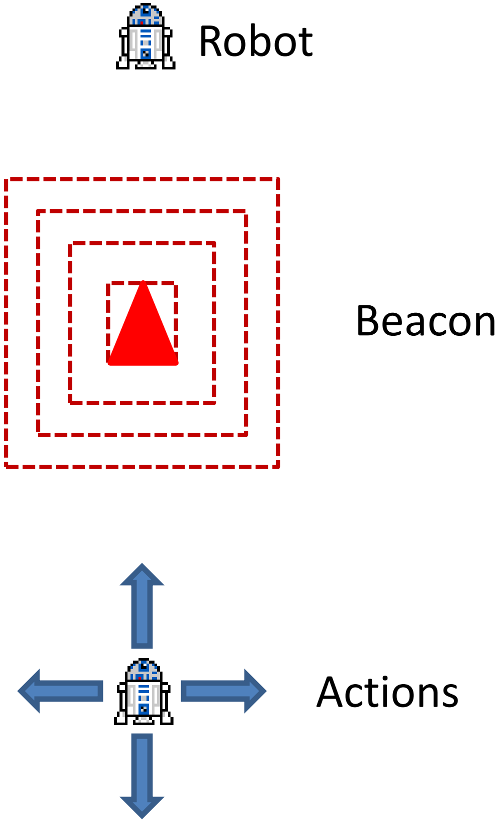
$$\widehat{p_t}[i] = \eta \Pr(o_t|S_t = i)\overline{p_t}[i]$$

3. Normalize belief (to get rid of η):

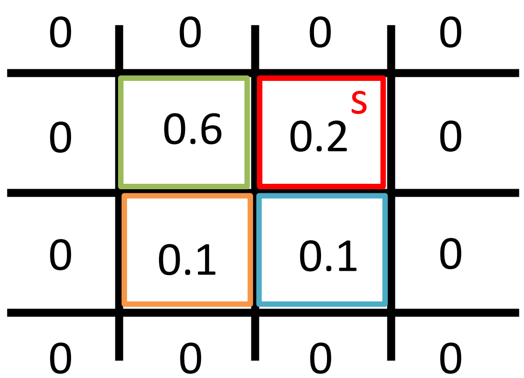
$$\widehat{p_t}[i] \leftarrow \frac{\widehat{p_t}[i]}{\eta}, \eta = \sum_{j=1}^{d} \widehat{p_t}[j]$$

Grid World





Prediction example



Robot model

"If told to perform an action, the robot will execute it with probability 90% or do nothing with probability 10%."

Action = "Move right!"

What is the probability of being in the red square? $\Pr(S_{t+1} = s | a_t = \text{right})?$

$$
\begin{aligned}
\Pr(S_{t+1} = s | a_t = \text{right}) \quad &= \Pr(S_{t+1} = s | S_t = s', a_t = \text{right}) \times \Pr(S_t = s') \\
&+ \Pr(S_{t+1} = s | S_t = s', a_t = \text{right}) \times \Pr(S_t = s') \\
&+ \Pr(S_{t+1} = s | S_t = s', a_t = \text{right}) \times \Pr(S_t = s') \\
&+ \Pr(S_{t+1} = s | S_t = s', a_t = \text{right}) \times \Pr(S_t = s')
\end{aligned}
$$

Prediction example

0	0	0	0
0	0.6	0.2 s	0
0	0.1	0.1	0
0	0	0	0

Robot model

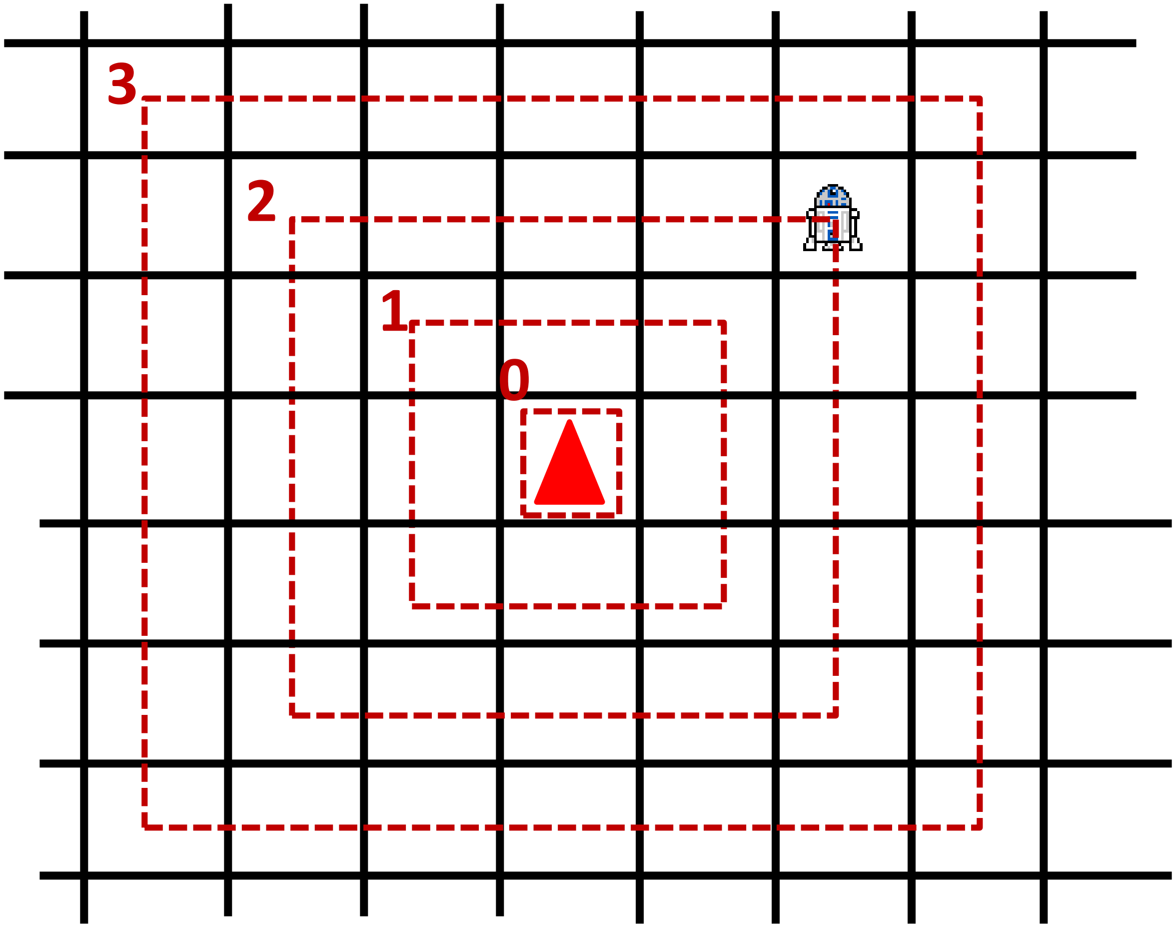
"If told to perform an action, the robot will execute it with probability 90% or do nothing with probability 10%."

Action = "Move right!"

What is the probability of being in the red square? $\Pr(S_{t+1} = s | a_t = \text{right})?$

$$\Pr(S_{t+1} = s | a_t = \text{right}) = 0.9 \times 0.6 + 0.0 \times 0.1 + 0.1 \times 0.2 + 0.0 \times 0.1 = 0.56$$

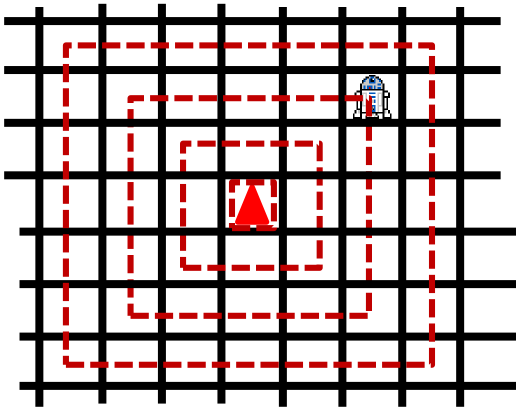
Sensor update



Beacon Model

- If at the beacon's location, returns 0 with probability 95%;
- If 1 square away, returns '1' with probability 70%;
- If 2 squares away, returns '2' with probability 80%;
- If 3 squares away, returns '3' with probability 60%;
- In all other cases, the beacon returns no reading ('-').

Sensor update



"What is my confidence about being in state *s* at time *t+1*, given that I took action *a* at time *t* and received observation *o* from beacon *b* at time *t+1*?"

$$\Pr(S_{t+1} = s | a_t = a, o_{t+1} = o_b)?$$

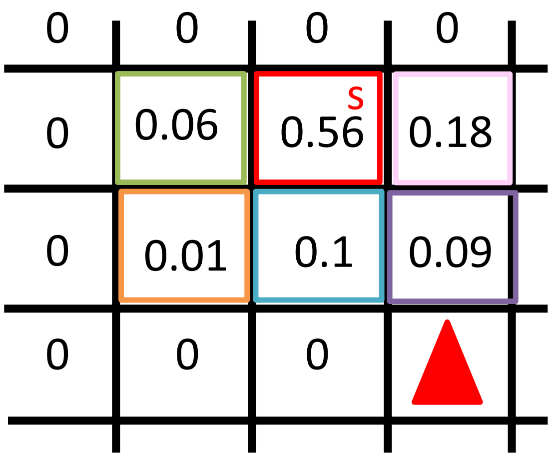
$$\Pr(S_{t+1} = s | a_t = a, o_{t+1} = o_b) \propto \Pr(o_{t+1} = o_b | S_{t+1} = s) \Pr(S_{t+1} = s | a_t = a)$$

Compute unnormalized probabilities.

$$\Pr(S_{t+1} = s | a_t = a, o_{t+1} = o_b) \leftarrow \frac{\Pr(S_{t+1} = s | a_t = a, o_{t+1} = o_b)}{\sum_{s'} \Pr(S_{t+1} = s' | a_t = a, o_{t+1} = o_b)}$$

Normalize their sum to 1.

Sensor update example



Prediction example. Go check it!

Action = "Move right!"

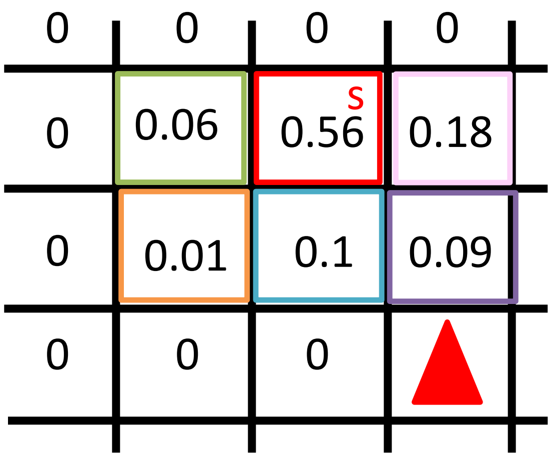
Observation = "You're 2 squares away."

Beacon Model

- If at the beacon's location, returns 0 with probability 95%;
- If 1 square away, returns '1' with probability 70%;
- If 2 squares away, returns '2' with probability 80%;
- If 3 squares away, returns '3' with probability 60%;
- In all other cases, the beacon returns no reading ('-').

$$\Pr(S_{t+1} = s | a_t = \text{right}, o'_{t+1} = 2) \quad \propto \Pr(o_{t+1} = 2 | s_{t+1} = s) \Pr(S_{t+1} = s | a_t = right)$$

Sensor update example



Prediction example. Go check it!

Action = “Move right!”

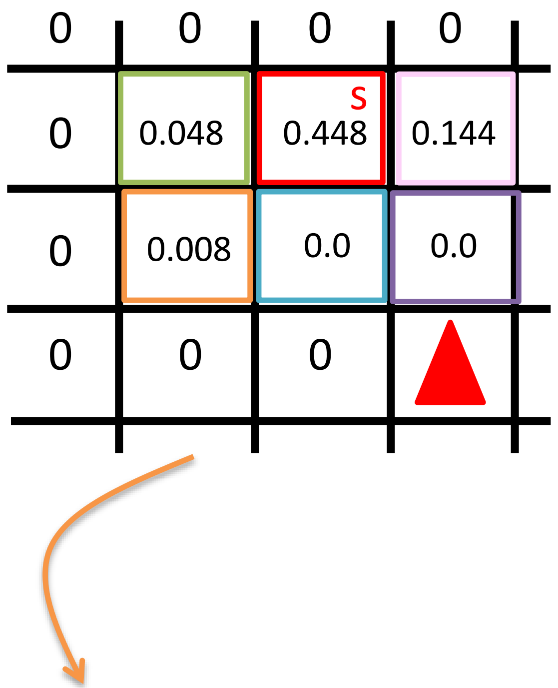
Observation = “You’re 2 squares away.”

Beacon Model

- If at the beacon’s location, returns 0 with probability 95%;
- If 1 square away, returns ‘1’ with probability 70%;
- If 2 squares away, returns ‘2’ with probability 80%;
- If 3 squares away, returns ‘3’ with probability 60%;
- In all other cases, the beacon returns no reading (‘-’).

$$\Pr(S_{t+1} = s | a_t = \text{right}, o'_{t+1} = 2) \quad \propto \Pr(o_{t+1} = 2 | S_{t+1} = s) \Pr(S_{t+1} = s | a_t = right)$$
$$= 0.8 \times 0.56$$
$$= 0.448$$

Sensor update example



Action = “Move right!”

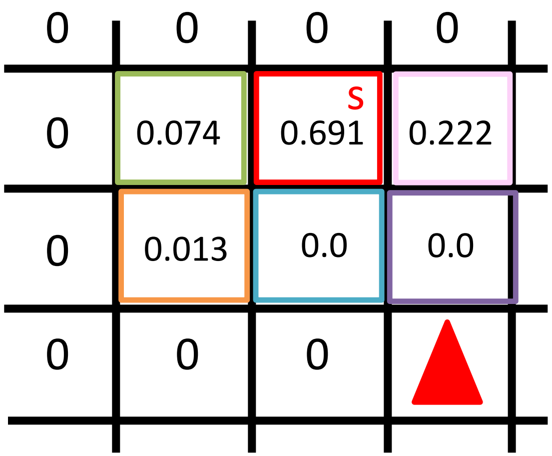
Observation = “You’re 2 squares away.”

Beacon Model

- If at the beacon’s location, returns 0 with probability 95%;
- If 1 square away, returns ‘1’ with probability 70%;
- If 2 squares away, returns ‘2’ with probability 80%;
- If 3 squares away, returns ‘3’ with probability 60%;
- In all other cases, the beacon returns no reading (‘-’).

Normalize the probabilities on the grid!

Sensor update example



Action = “Move right!”

Observation = “You’re 2 squares away.”

Beacon Model

- If at the beacon’s location, returns 0 with probability 95%;
- If 1 square away, returns ‘1’ with probability 70%;
- If 2 squares away, returns ‘2’ with probability 80%;
- If 3 squares away, returns ‘3’ with probability 60%;
- In all other cases, the beacon returns no reading (‘-’).

Done with belief state updates!

3. RAO*

"It's a trap!"

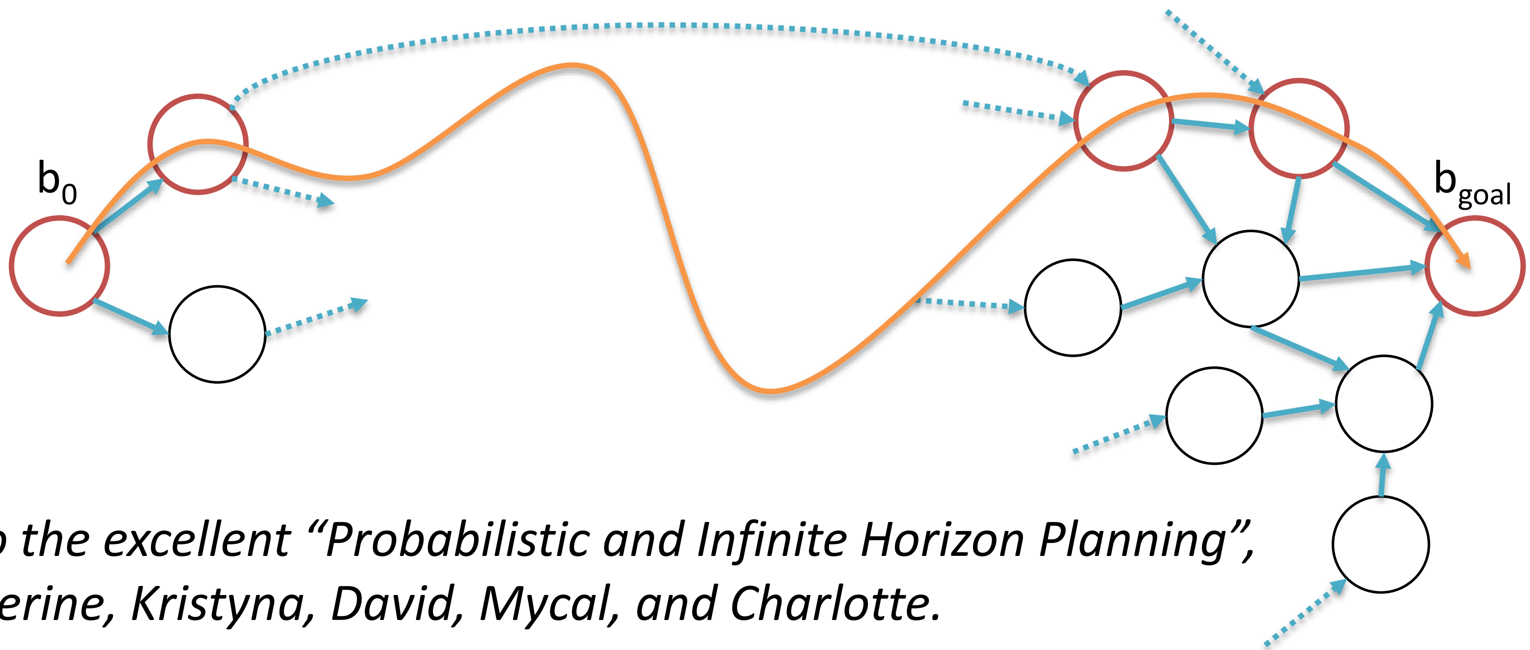
—RAO*, after determining that a policy was too risky.

RAO* = AO* + Belief states + Execution risk

"Probability of violating constraints $\mathbb{c}$ during execution." $\leq \Delta \quad \equiv \quad er(b_0, \mathbb{c}|\pi) \leq \Delta$

Key 1: (admissible) value heuristic guiding search towards "promising" policies;

Key 2: (admissible) *execution risk* heuristic allowing risk bounds to be propagated forward.



See also the excellent "Probabilistic and Infinite Horizon Planning", by Katherine, Kristyna, David, Mycal, and Charlotte.

Elements of a CC-POMDP

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

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

$\mathbb{O}$: discrete set of observations.

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

O: $\mathbb{S}\times\mathbb{O}\rightarrow [0,1]$, *observation function*

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

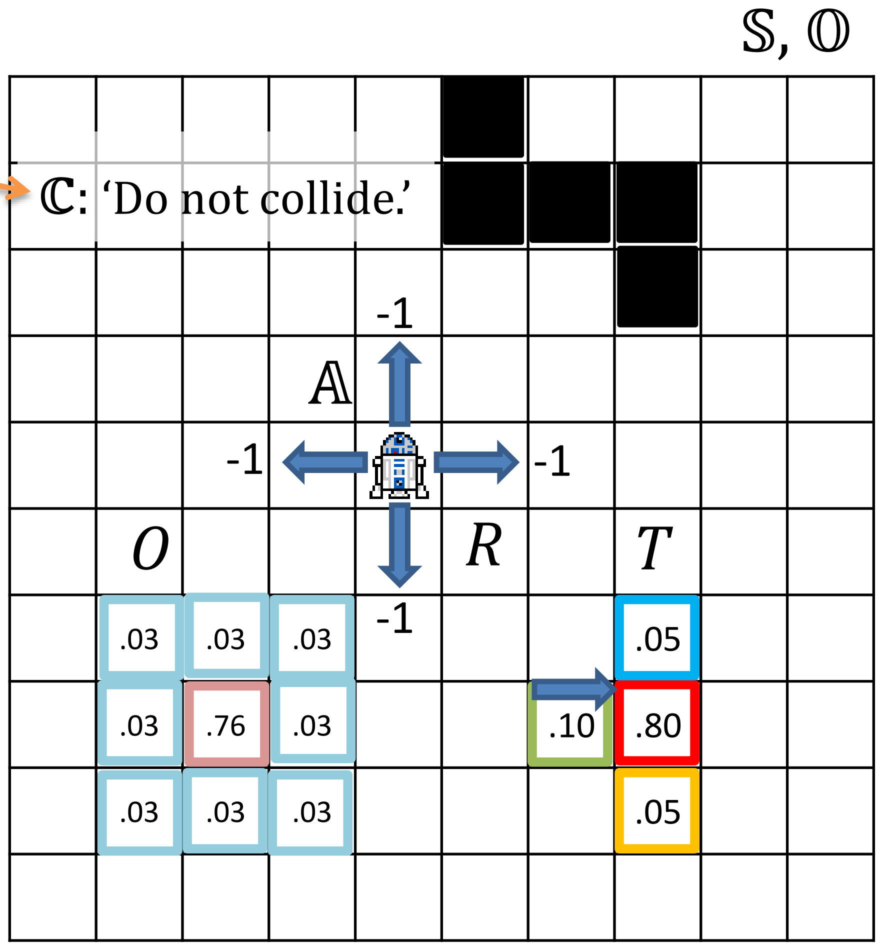
$\mathbb{C}$: set of state constraints.

Δ: risk bound.

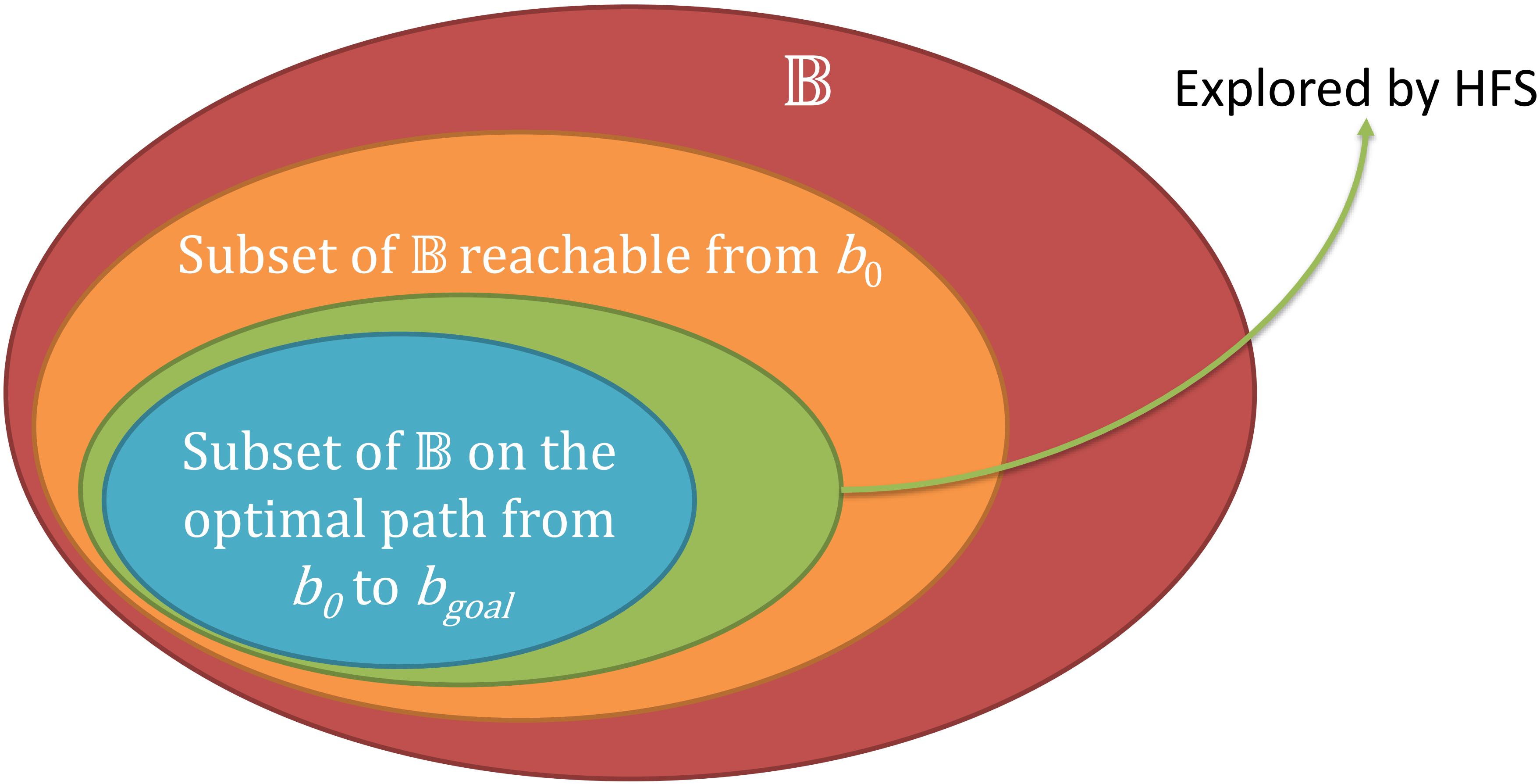
$$T(s_k, a_k, s_{k+1}) = \Pr(s_{k+1}|s_k, a_k)$$

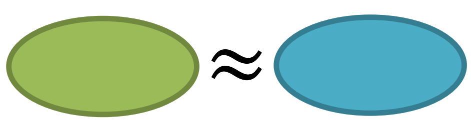
$$O(s_k, o_k) = \Pr(o_k|s_k)$$

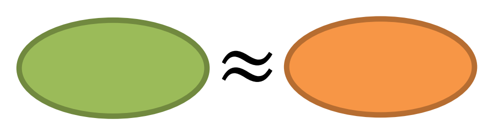
Δ=0.01→ 'Collision probability must be less than 1%.'



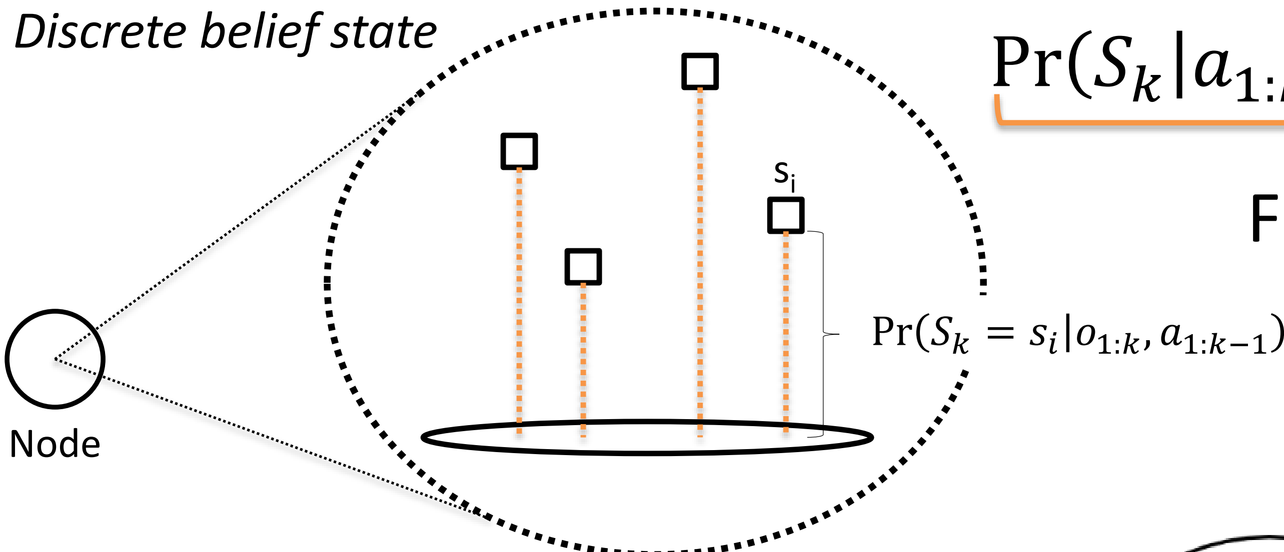
Searching from an initial belief b_0



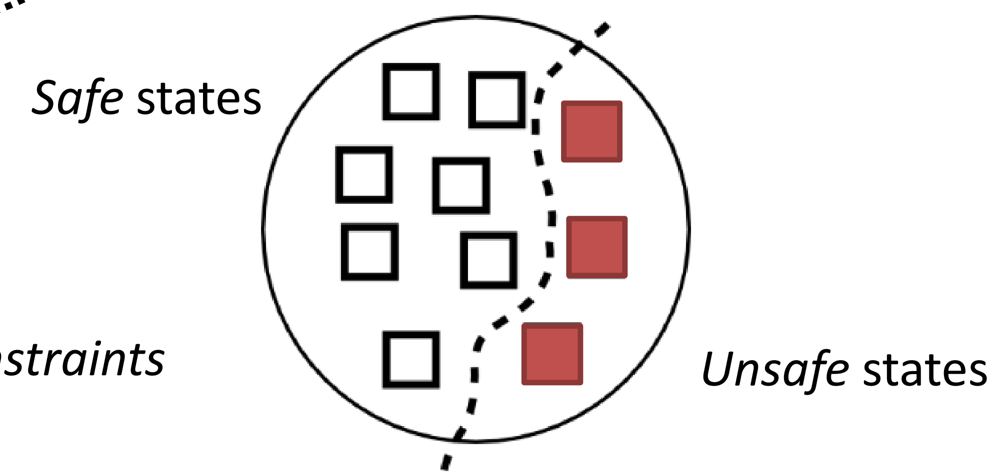
Good heuristic: 

Bad heuristic: 

RAO* nodes are belief states



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

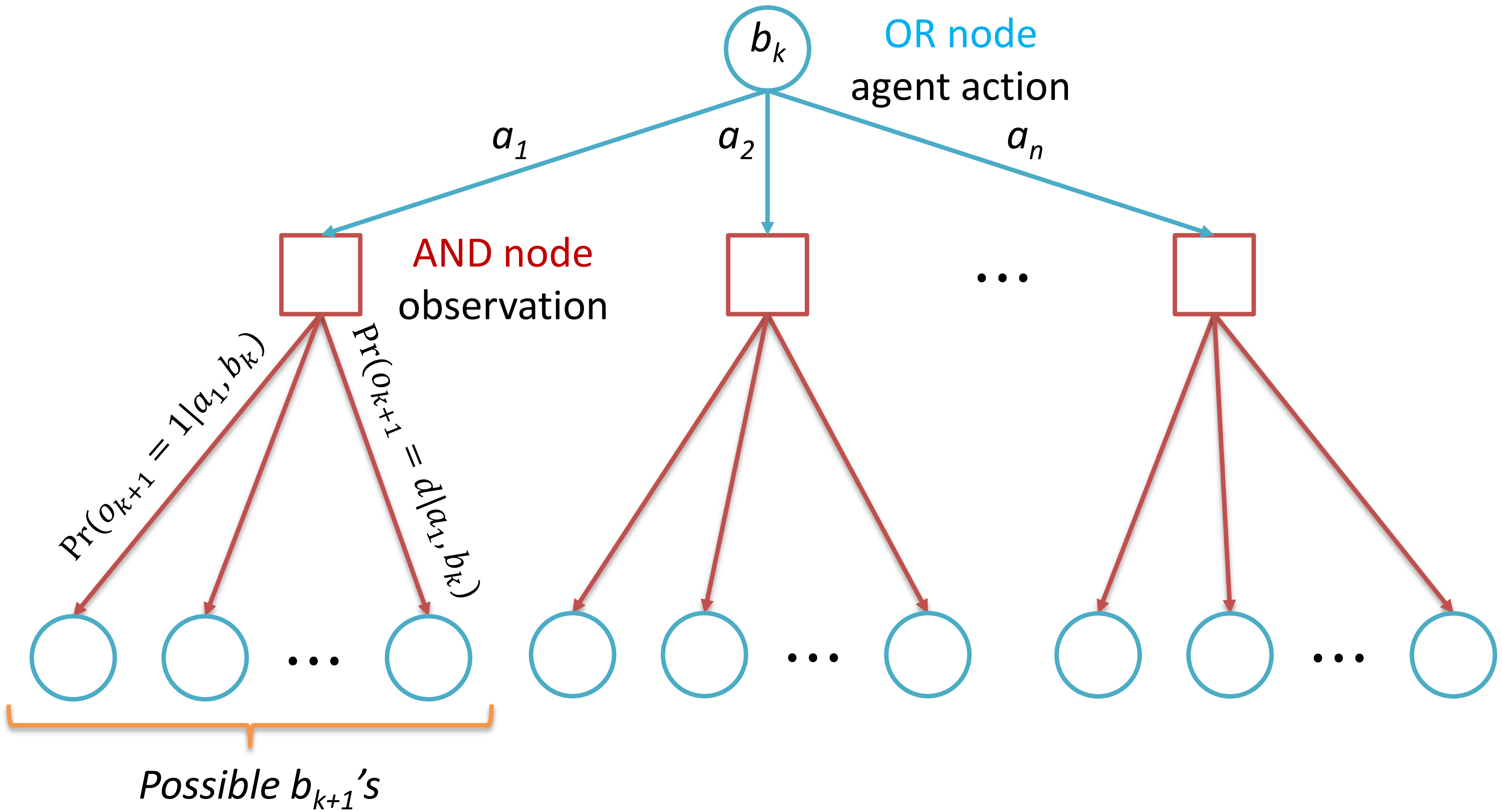


State is "safe"

$\Leftrightarrow$

itself and its ancestors do not violate constraints

Partially observable AND-OR search

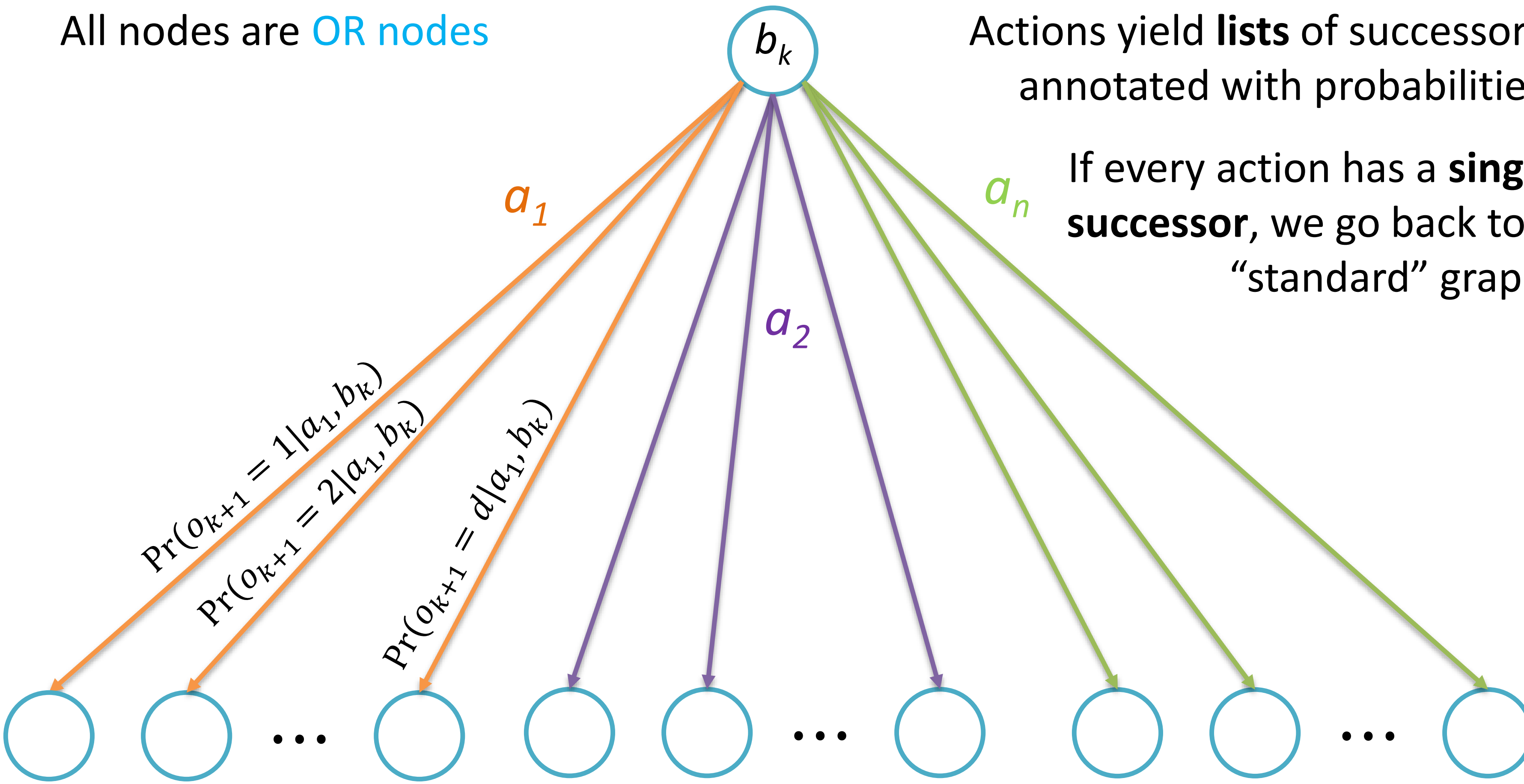


Hypergraph representation

All nodes are OR nodes

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

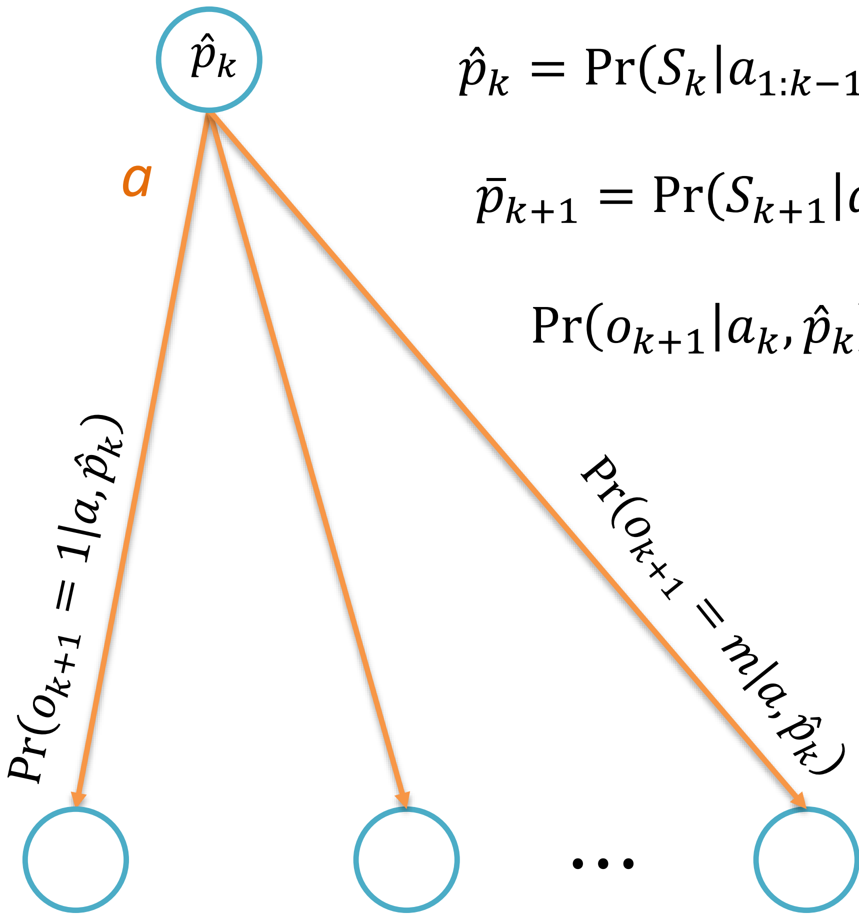
If every action has a **single successor**, we go back to a "standard" graph.



Key concepts

1. How to compute transition probabilities
2. How to compute (admissible) utility estimates
3. How to compute (admissible) execution risk estimates
4. How to put them all together

3.1 Hyperedge probabilities



$\Pr(S_{k+1} | a_{1:k}, o_{1:k}, o_{k+1} = 1)$ $\Pr(S_{k+1} | a_{1:k}, o_{1:k}, o_{k+1} = m)$ (filtering)

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

$$\bar{p}_{k+1} = \Pr(S_{k+1} | a_{1:k}, o_{1:k})$$

(prediction)

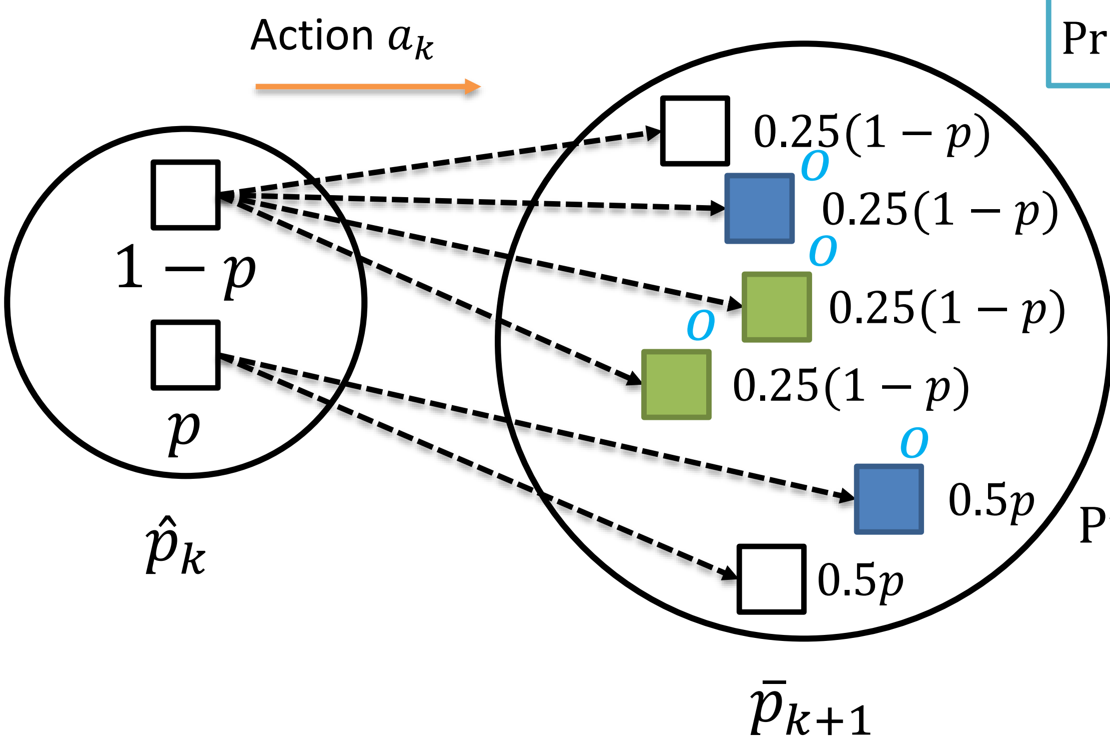
$$\Pr(o_{k+1} | a_k, \hat{p}_k) = \sum_{s_{k+1}} \underbrace{\Pr(o_{k+1} | s_{k+1})}_{\text{Observation model}} \bar{p}_{k+1}(s_{k+1})$$

Computing hyperedge probabilities

$$\bar{p}_{k+1} = \Pr(S_{k+1}|a_{1:k}, o_{1:k}) = T(a_k)\hat{p}_k \quad \Pr(o_{k+1}|a_k, \hat{p}_k) = \sum_{s_{k+1}} \Pr(o_{k+1}|s_{k+1})\, \bar{p}_{k+1}(s_{k+1})$$

$\Pr(o|■) = 0.8$
$\Pr(-|■) = 0.2$

$\Pr(o|■) = 0.6$
$\Pr(\times|■) = 0.4$



$\Pr(o|a_k, \hat{p}_k)$?

$$\begin{aligned}\Pr(o|a_k, \hat{p}_k) &= 0.25(1-p) \times 0.8 \\ &+ 0.25(1-p) \times 0.6 \\ &+ 0.25(1-p) \times 0.6 \\ &+ 0.5p \times 0.8\end{aligned}$$

3.2 Admissible utility estimates

Unknown

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

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

Admissible ("optimistic") estimate of future reward.

Should be "easy" to compute.

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

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

3.3 Execution risk

$Sa_i = 1$: *agent hasn't violated* $\mathbb{c}$ until i-th step

$$\underbrace{er(b_k, \mathbb{c}|\pi)}_{\text{Probability of violating constraints from } k \text{ onwards.}} = 1 - \underbrace{\Pr\left(\bigwedge_{i=k}^{T} Sa_i | b_k, \pi\right)}_{\text{Probability of remaining safe from } k \text{ onwards.}}$$

(Ono et al., 2012)

$$er(b_k|\pi) = \underbrace{r_b(b_k)}_{\text{Immediate risk at the current belief state.}} + (1 - r_b(b_k)) \sum_{o_{k+1}} \underbrace{Pr^{sa}(o_{k+1}|\pi(b_k), b_k)}_{\text{Observations originated from safe states.}} er(b_{k+1}|\pi)$$

Execution risk

$$er(b_k|\pi) = \underbrace{r_b(b_k)}_{\text{Immediate risk at the current belief state.}} + (1 - r_b(b_k)) \sum_{o_{k+1}} \underbrace{Pr^{sa}(o_{k+1}|\pi(b_k), b_k)}_{\text{Observations originated from safe states.}} er(b_{k+1}|\pi)$$

$$r_b(b_k) = \sum_{s_k \in \mathbb{S}} b(s_k) \underbrace{c_v(p(s_k), \mathbb{C})}$$

1 if the sequence of states ("path") leading to s_k violates $\mathbb{C}$

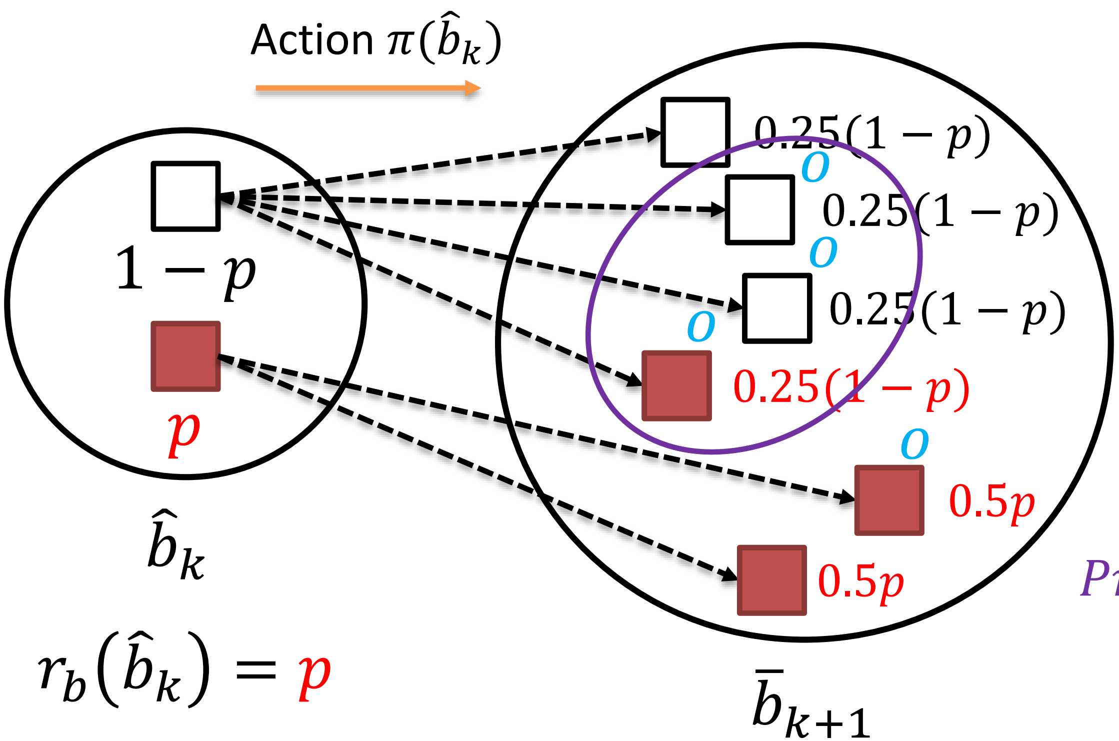
$$\bar{b}^{sa}(s_{k+1}|a_k) = \Pr(s_{k+1}|Sa_k, a_k, b_k) = \frac{\sum_{s_k: c_v(p(s_k),\mathbb{C})=0} T(s_k, a_k, s_{k+1}) b(s_k)}{1 - r_b(b_k)}$$

$$\Pr^{sa}(o_{k+1}|a_k, b_k) = \Pr(o_{k+1}|Sa_k, a_k, b_k) = \sum_{s_{k+1}} O(s_{k+1}, o_{k+1})\, \bar{b}^{sa}(s_{k+1}|a_k)$$

Execution risk in pictures

$$er(b_k|\pi) = r_b(b_k) + (1 - r_b(b_k)) \sum_{o_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k), b_k) \underbrace{er(b_{k+1}|\pi)}$$

What about this term?



$r_b(\hat{b}_k) = p$

$$Pr^{sa}(o|\cdot) = \frac{3 \times 0.25(1-p)}{1-p}$$
$$= 0.75$$

Estimating execution risk

$$er(b_k|\pi) = r_b(b_k) + (1 - r_b(b_k)) \sum_{o_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k), b_k) er(b_{k+1}|\pi)$$

Admissible ("optimistic") estimate of future execution risk

$$h_{er}(b_{k+1}|\pi) \leq er(b_{k+1}|\pi)$$

Should be "easy" to compute.

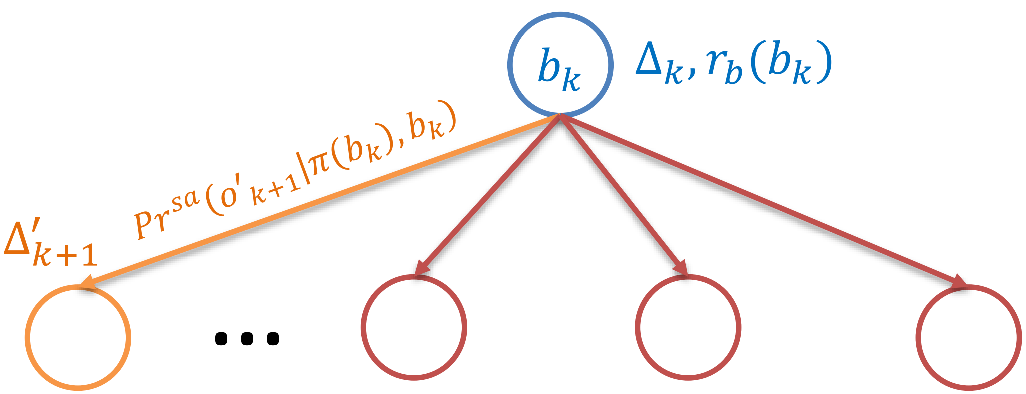
$$\boxed{er_h(b_k|\pi) = r_b(b_k) + (1 - r_b(b_k)) \sum_{o_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k), b_k) h_{er}(b_{k+1}|\pi)}$$

$h_{er}(b_{k+1}|\pi) = r_b(b_{k+1})$ is always admissible

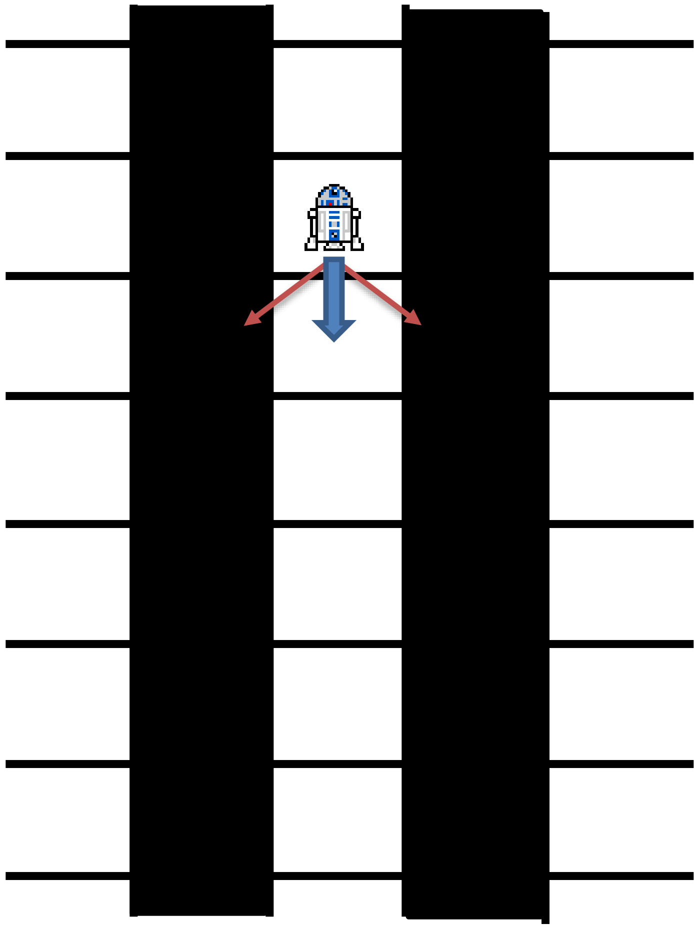
Propagating execution risk estimates

$$er_h(b_k|\pi) = r_b(b_k) + (1 - r_b(b_k)) \sum_{o_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k), b_k) h_{er}(b_{k+1}|\pi)$$

$$\Delta'_{k+1} = \frac{1}{Pr^{sa}(o'_{k+1}|\pi(b_k), b_k)} \left(\frac{\Delta_k - r_b(b_k)}{1 - r_b(b_k)} - \sum_{o_{k+1} \neq o'_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k), b_k) h_{er}(b_{k+1}|\pi) \right)$$

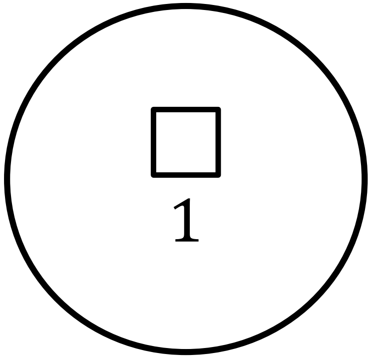


Risk example

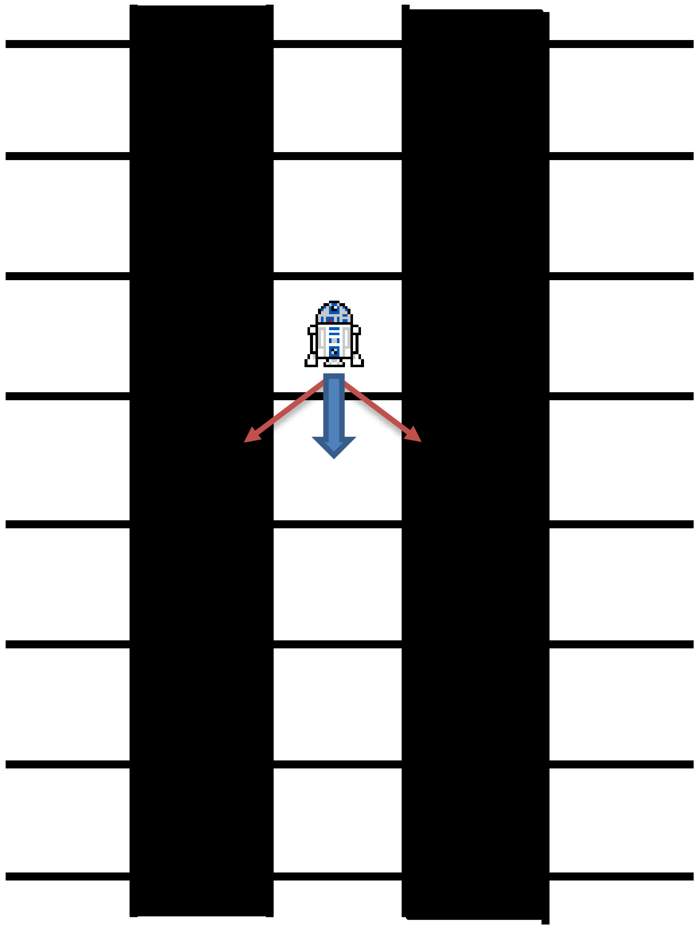


Robot model

"If told to move, R2D2 achieves the desired cell with probability 90%, or slips to either side with probability 5%."

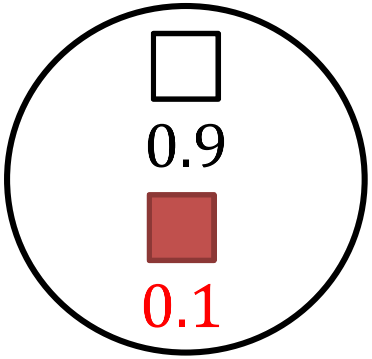


Risk example



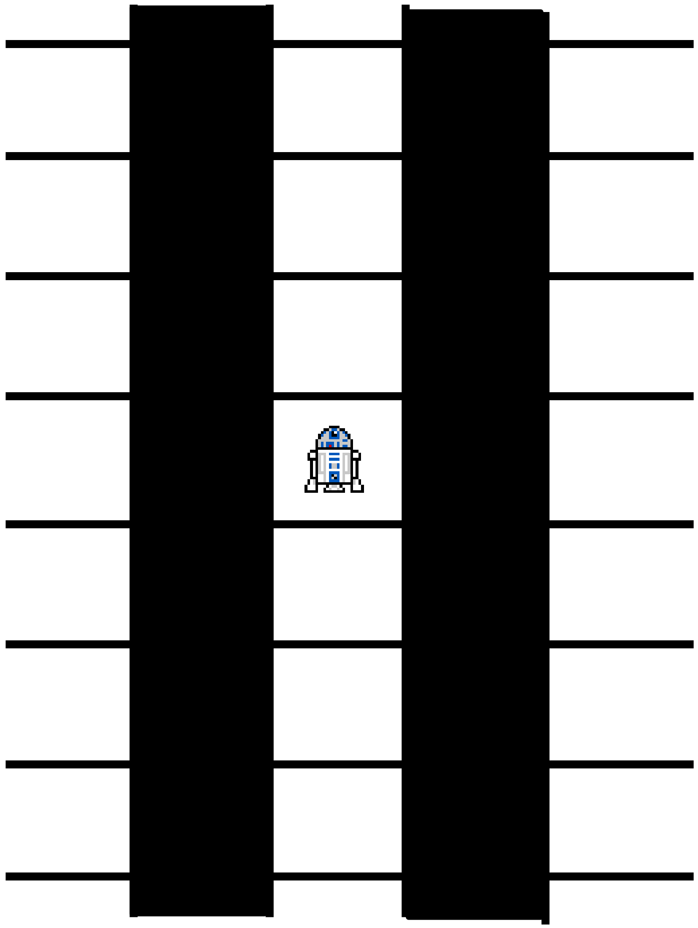
Robot model

"If told to move, R2D2 achieves the desired cell with probability 90%, or slips to either side with probability 5%."



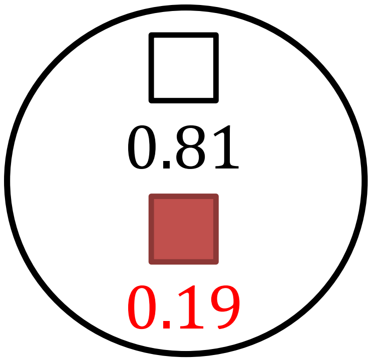
Both particles share the same position

Risk example



Robot model

"If told to move, R2D2 achieves the desired cell with probability 90%, or slips to either side with probability 5%."



Both particles share the same position

3.4 RAO* in a nutshell

Additions to AO shown in red*

- *Input*: implicit partially observable AND-OR search problem

$$< \mathbb{S}, \mathbb{A}, \mathbb{O}, T, O, R, \mathbb{C}, \Delta >, b_0$$

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

- *Strategy*: incrementally build solutions forward from b_0, using h to estimate future utilities (just like A*!) and h_{er} to estimate policy risk. 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.

RAO*'s pseudocode

Additions to AO shown in red*

Input: $< \mathbb{S}, \mathbb{A}, \mathbb{O}, T, O, R, \mathbb{c}, \Delta >, b_0$

Output: Policy π: $\mathbb{B} \rightarrow \mathbb{A}$

Heuristics for
utility and execution risk

Explicit graph $G \leftarrow b_0$, $g \leftarrow$ Best partial policy of G

while best partial policy graph g has nonterminal leafs

$m \leftarrow$ Expand any nonterminal leaf from g

Add to G the children in m which do not violate risk bound

Bellman
backups

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

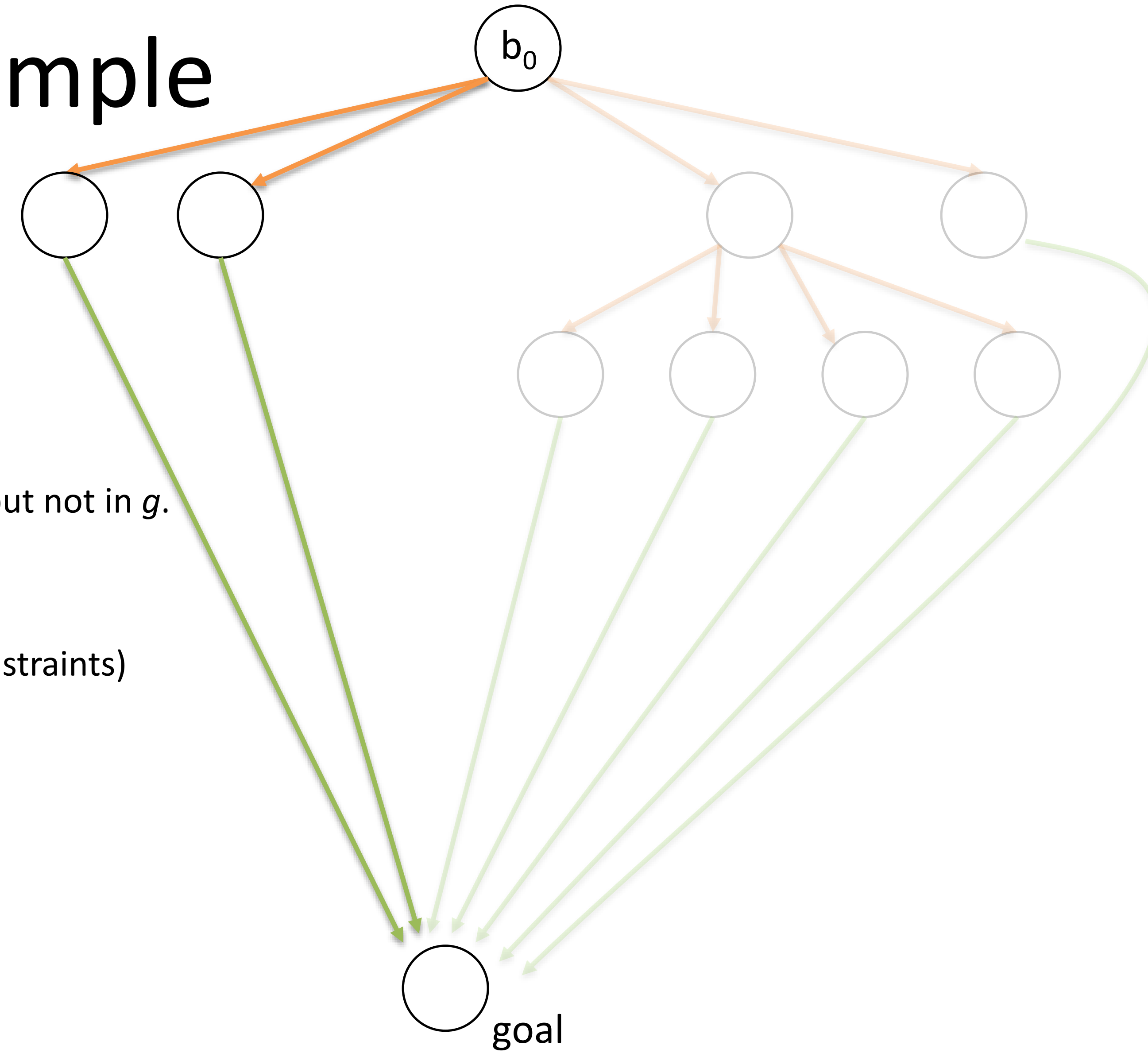
Update utility and execution risk for n

$\pi \leftarrow$ Choose best action at n not violating risk bound

Update g with the new π

g is the graph obtained by
following π from b_0

RAO* example



Node in *g*, the greedy graph.

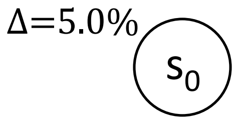
Node in *G*, the explicit graph, but not in *g*.

Node with $r_b = 1$ (violates constraints)

Known value

Heuristic estimate

Start

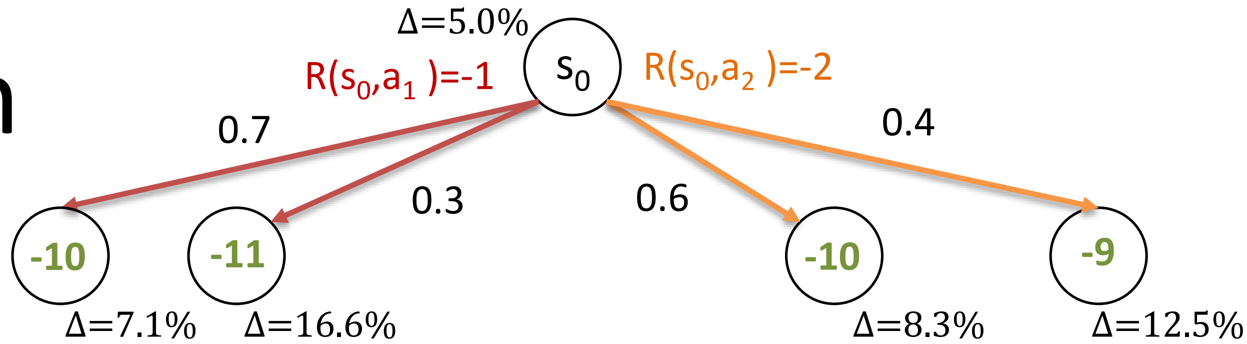


Open nodes: [s_0]

g={s_0: None }

G starts just with just the initial state s_0

Expansion



Open nodes: [s_0]

g={s_0: None }

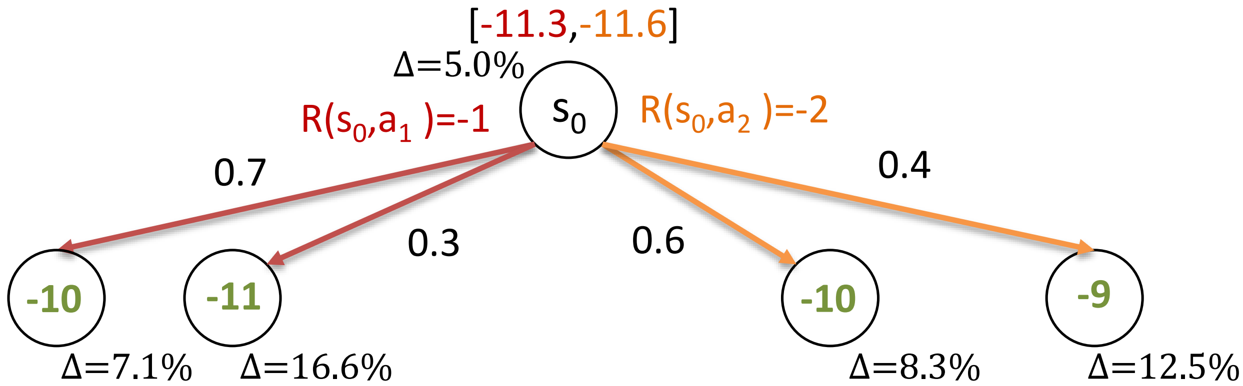
1. *Choose an open node to expand $\rightarrow s_0$*
2. *Estimate the value and execution risk of leaf nodes*
3. *Propagate risk bounds*

$h_{er} = r_b = 0$ for all leaves

$$\Delta'_{k+1} = \frac{1}{Pr^{sa}(o'_{k+1}|\pi(b_k), b_k)}\left(\frac{\Delta_k - r_b(b_k)}{1 - r_b(b_k)} - \sum_{o_{k+1} \neq o'_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k), b_k) h_{er}(b_{k+1}|\pi)\right)$$

$$= \frac{1}{0.6}\left(\frac{0.05 - 0}{1 - 0} - 0.4 \times 0\right) = 8.3\%$$

Backup



Open nodes: [s_0]

g={s_0: None }

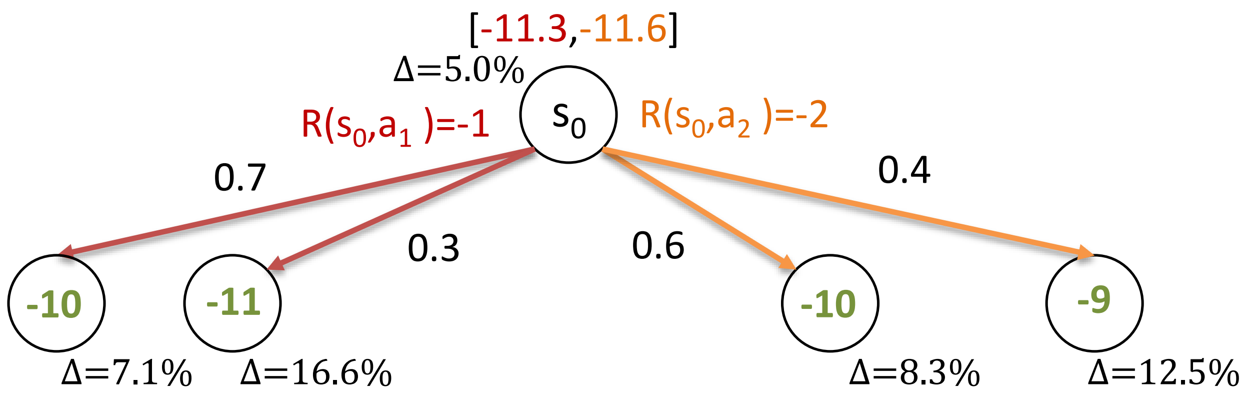
3. Backup value and execution risk for the currently expanded node (s_0) and all its ancestors that are part of g (no ancestors), recording the best value at each node.

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

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

$$er(s_0,a_1) = er(s_0,a_2) = 0 < 5\%$$

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

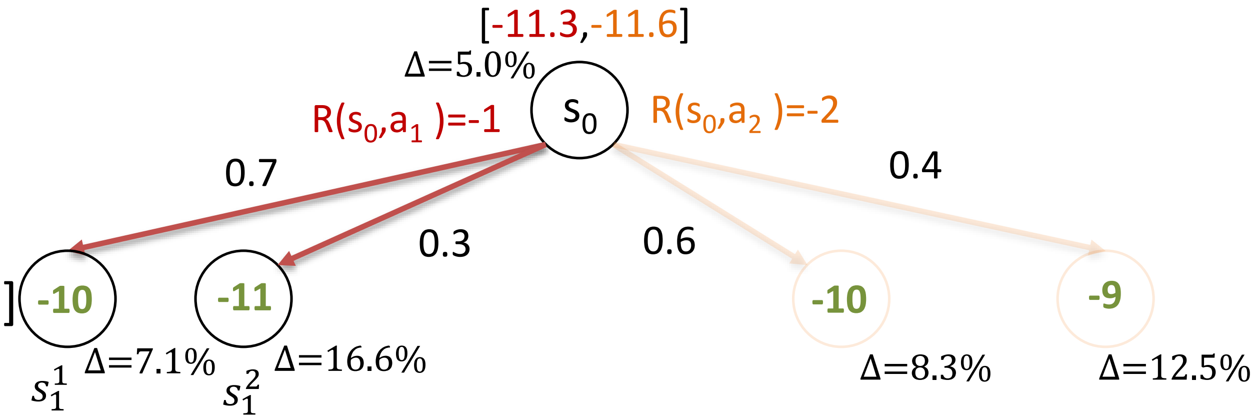
$$\boxed{V_h(s_0, a_1) = -1 + (-10 * 0.7 - 11 * 0.3) = -11.3}$$

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

Update g

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

$g=\{s_0\text{: } a1\}$

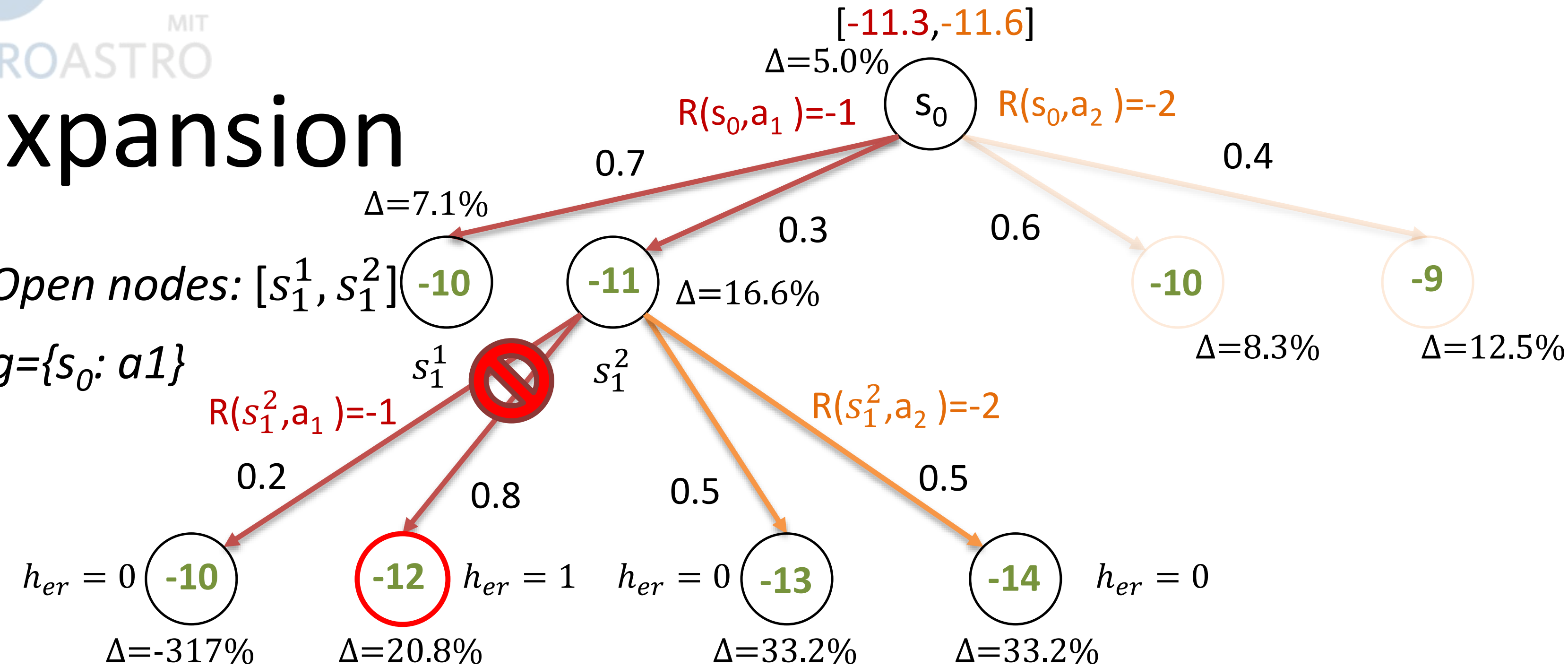


4. Update g and the list of open nodes (non-terminal) by selecting the best action at the nodes which got their values updated

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

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

Expansion



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

g={s_0: a1}

$$\frac{1}{0.2}\left(\frac{0.166-0}{1-0}-0.8\times 1\right)=-317\% \qquad \frac{1}{0.8}\left(\frac{0.166-0}{1-0}-0.2\times 0\right)=20.8\%$$

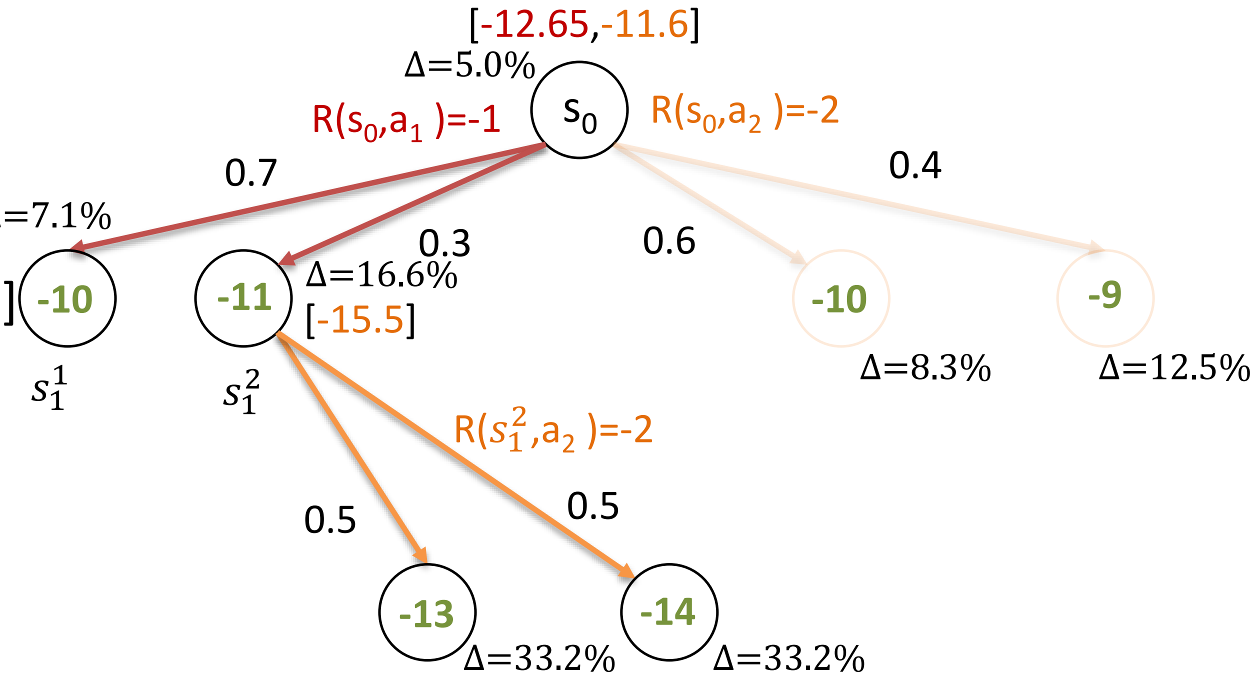
$$\Delta'_{k+1}=\frac{1}{Pr^{sa}(o'_{k+1}|\pi(b_k),b_k)}\left(\frac{\Delta_k-r_b(b_k)}{1-r_b(b_k)}-\sum_{o_{k+1}\neq o'_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k),b_k)h_{er}(b_{k+1}|\pi)\right)$$

1. *Choose any open node to expand* $\rightarrow s_1^2$
2. *Estimate the value and execution risk of leaf nodes*
3. *Propagate risk bounds*

Backup

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

g={s_0: a1}



3. Backup values and execution risk 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.

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

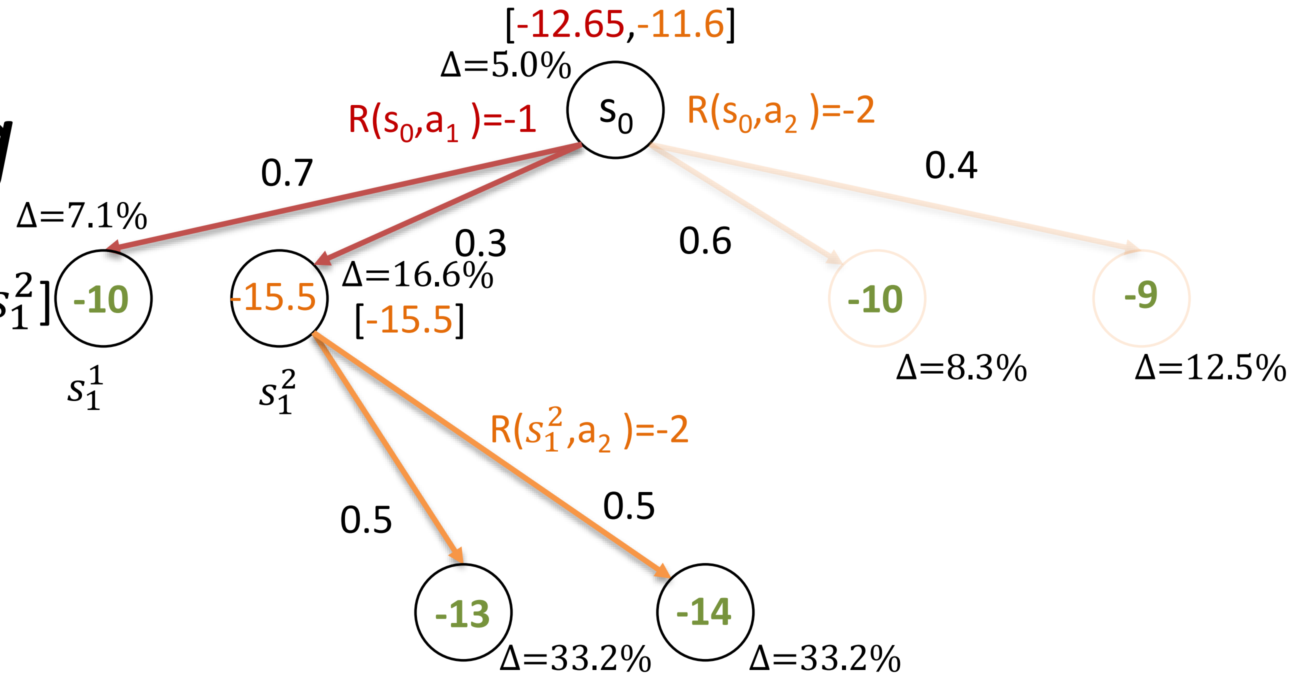
$$V_h(s_0, a_1) = -1 + (-10 * 0.7 - 15.5 * 0.3) = -12.65$$

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

$$er(s_1^2, a_2) = 0 < 16.6\%, er(s_0, a_2) = 0 < 5\%$$

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.

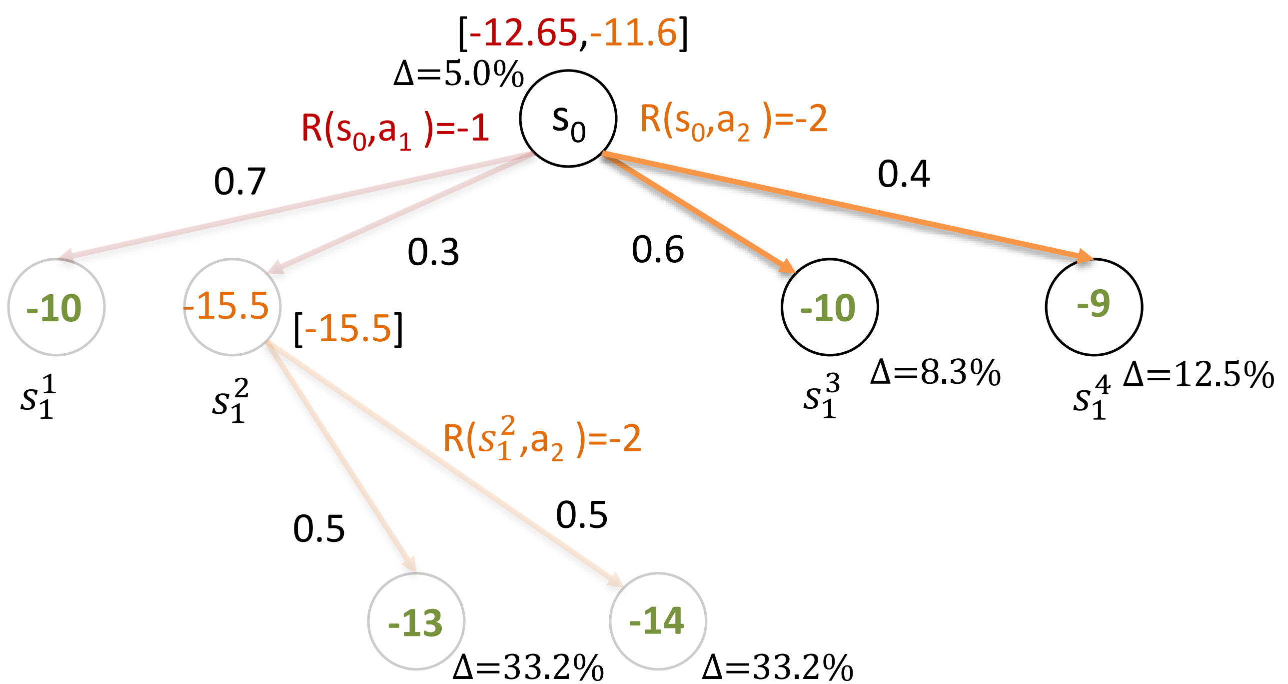
$$\boxed{V_h(s_1^2, a_2) = -2 + (-13 * 0.5 - 14 * 0.5) = -15.5}$$

$$V_h(s_0, a_1) = -1 + (-10 * 0.7 - 15.5 * 0.3) = -12.65$$

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

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.

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

$$V_h(s_0, a_1) = -1 + (-10 * 0.7 - 15.5 * 0.3) = -12.65$$

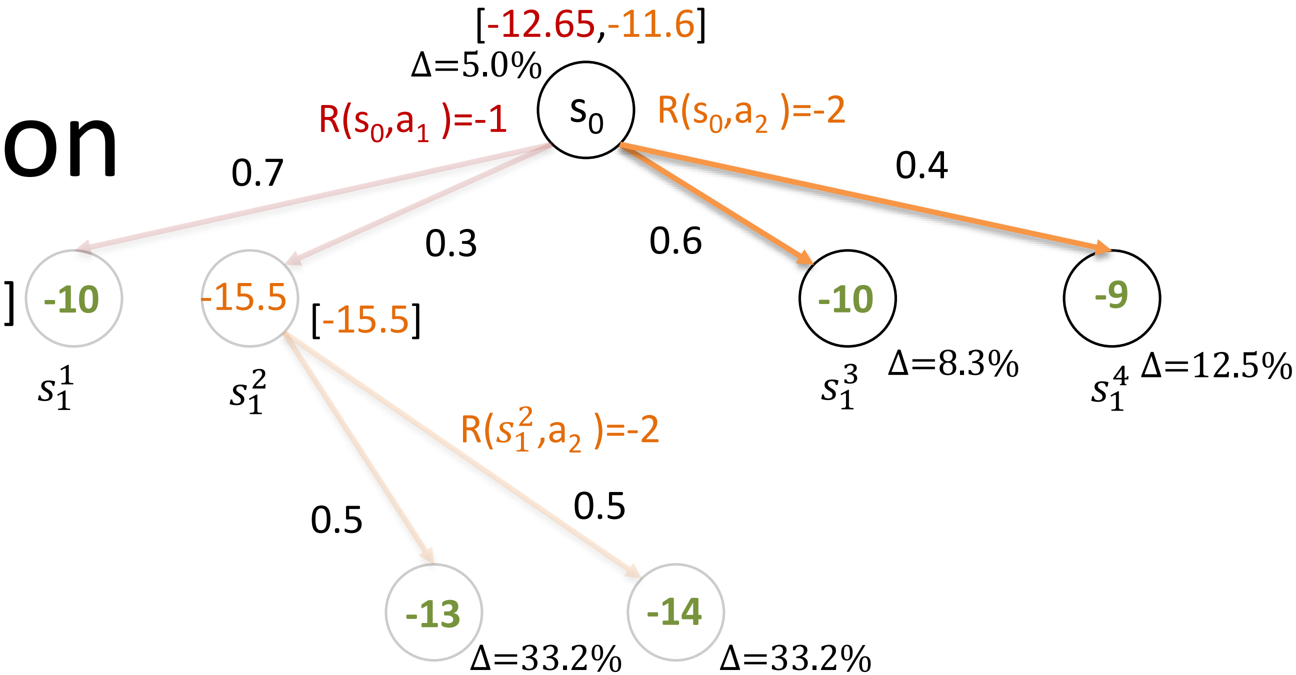
$$\boxed{V_h(s_0, a_2) = -2 + (-10 * 0.6 - 9 * 0.4) = -11.6}$$

From leafs to the root

Termination

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

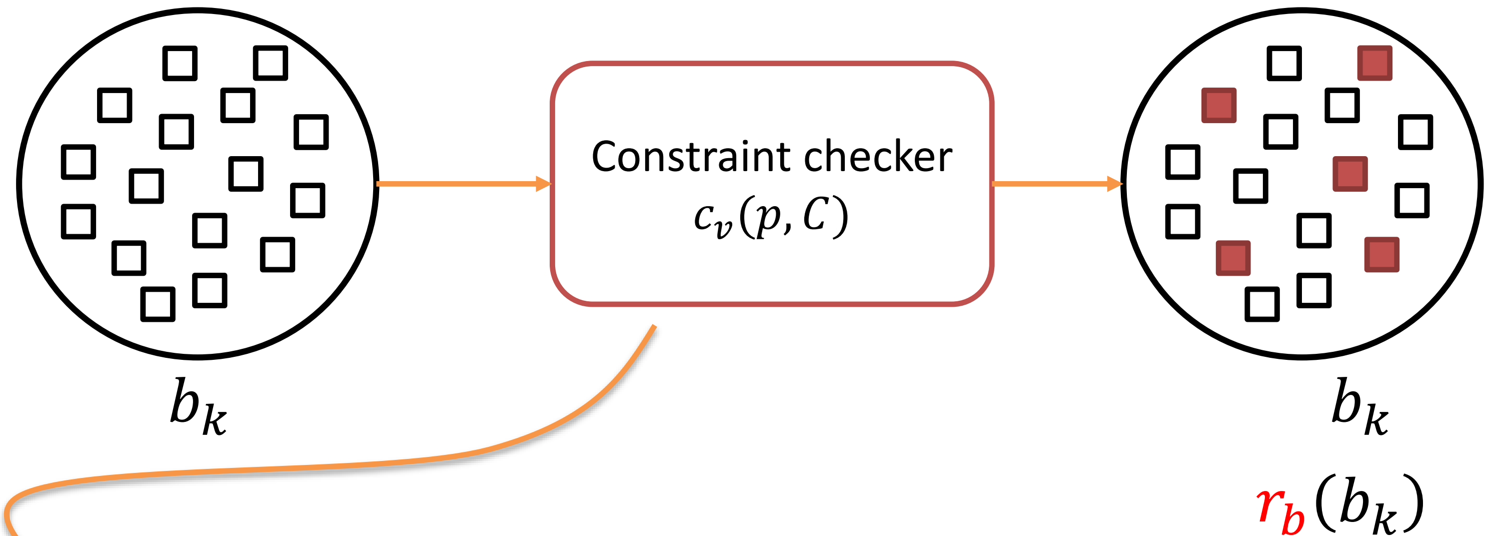
g={s_0: a2}



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.

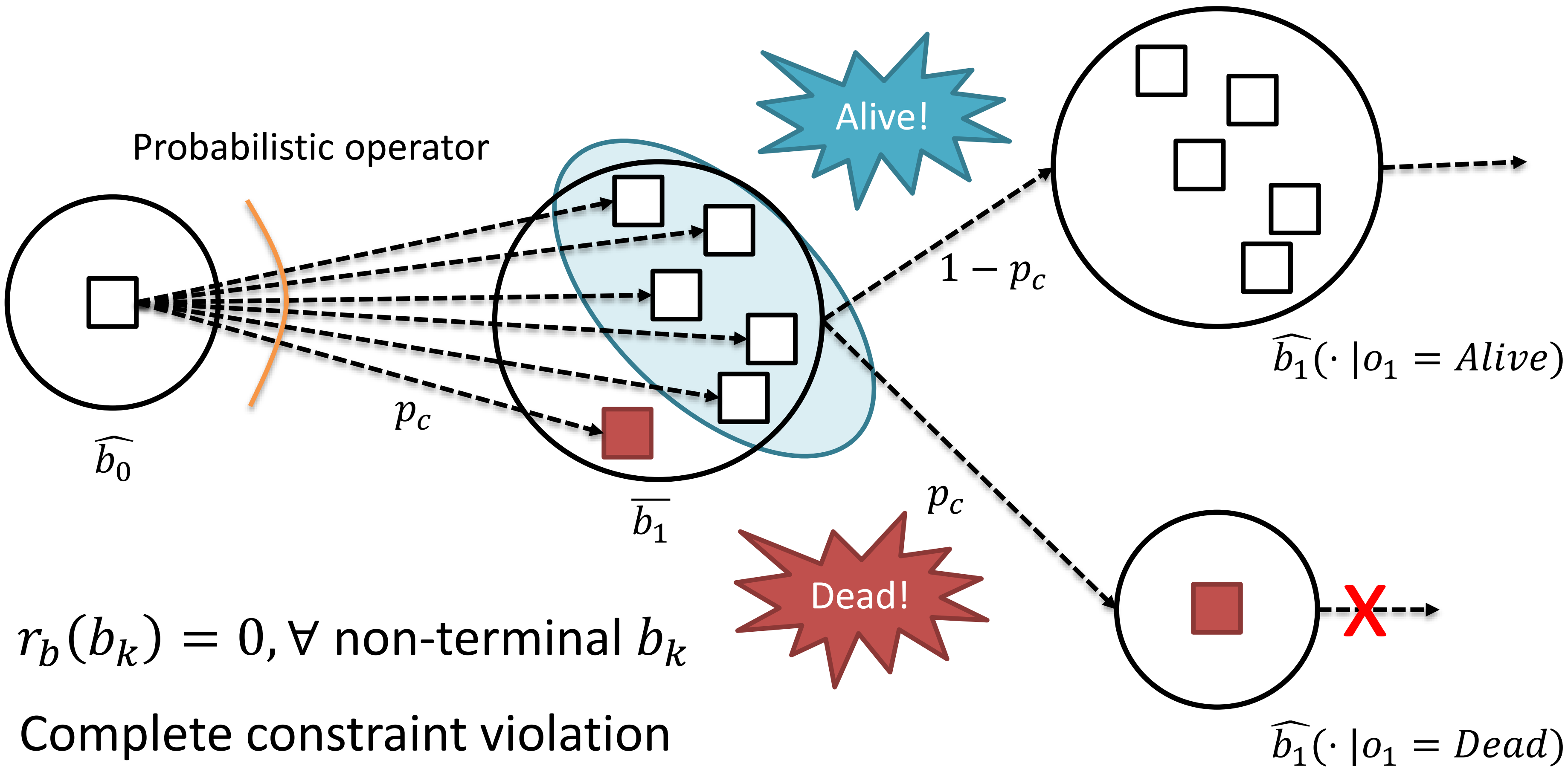
How hard is it to assess risk in RAO*?



Must run *really fast* for RAO* to be useful in practice!

e.g., see the probabilistic scheduling lecture by Andrew Wang and Cheng Fang.

Terminal violations = Observable violations

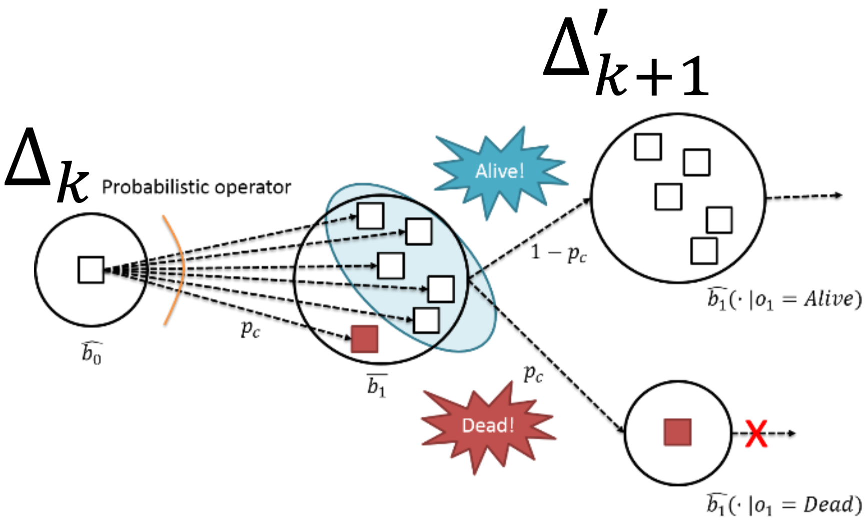


$r_b(b_k) = 0, \forall$ non-terminal b_k

Complete constraint violation awareness!

Dynamic vs Static risk allocation

$$\Delta'_{k+1} = \frac{1}{Pr^{sa}(o'_{k+1}|\pi(b_k), b_k)}\left(\frac{\Delta_k - r_b(b_k)}{1 - r_b(b_k)} - \sum_{o_{k+1} \neq o'_{k+1}} Pr^{sa}(o_{k+1}|\pi(b_k), b_k) h_{er}(b_{k+1}|\pi)\right)$$



Static allocation: $\Delta'_{k+1} = \Delta_k - p_c$

Dynamic allocation: $\Delta'_{k+1} = \frac{1}{1 - p_c}\left(\frac{\Delta_k - 0}{1 - 0} - p_c \cdot 1\right) = \frac{\Delta_k - p_c}{1 - p_c}$

Some takeaways

1. *Execution risk* should be applicable to risk-aware planning in general and could be incorporated into other POMDP solvers to endow them with a keen sensitivity to risk;

2. Risk-bounded plan execution improves upon the conservatism of risk-minimal alternatives while offering strict safety guarantees;

3. Efficient risk-aware constraint solvers are necessary for risk-aware planning;