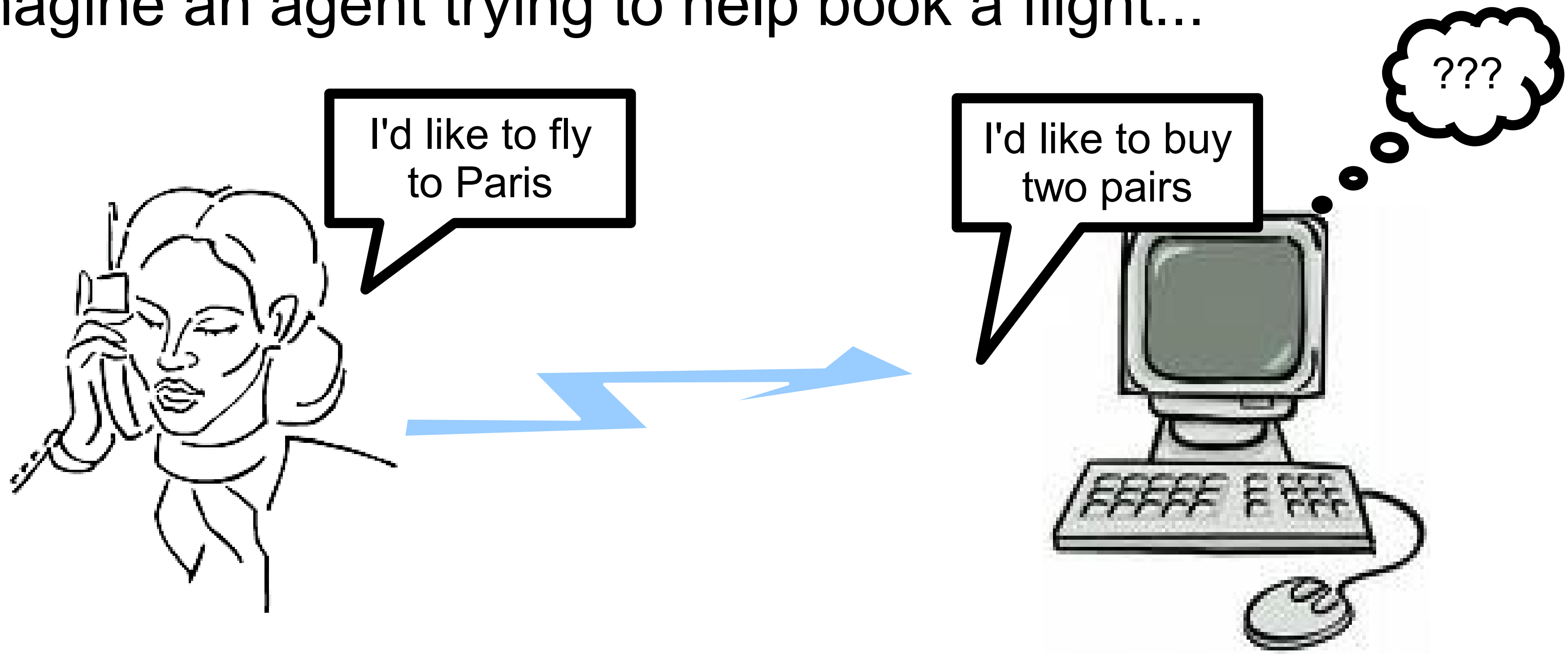
The Permutable POMDP: Fast Solutions to POMDPs for Preference Elicitation

Finale Doshi
MIT, Cambridge University
Nick Roy
MIT

CSAIL

Motivation

Imagine an agent trying to help book a flight...

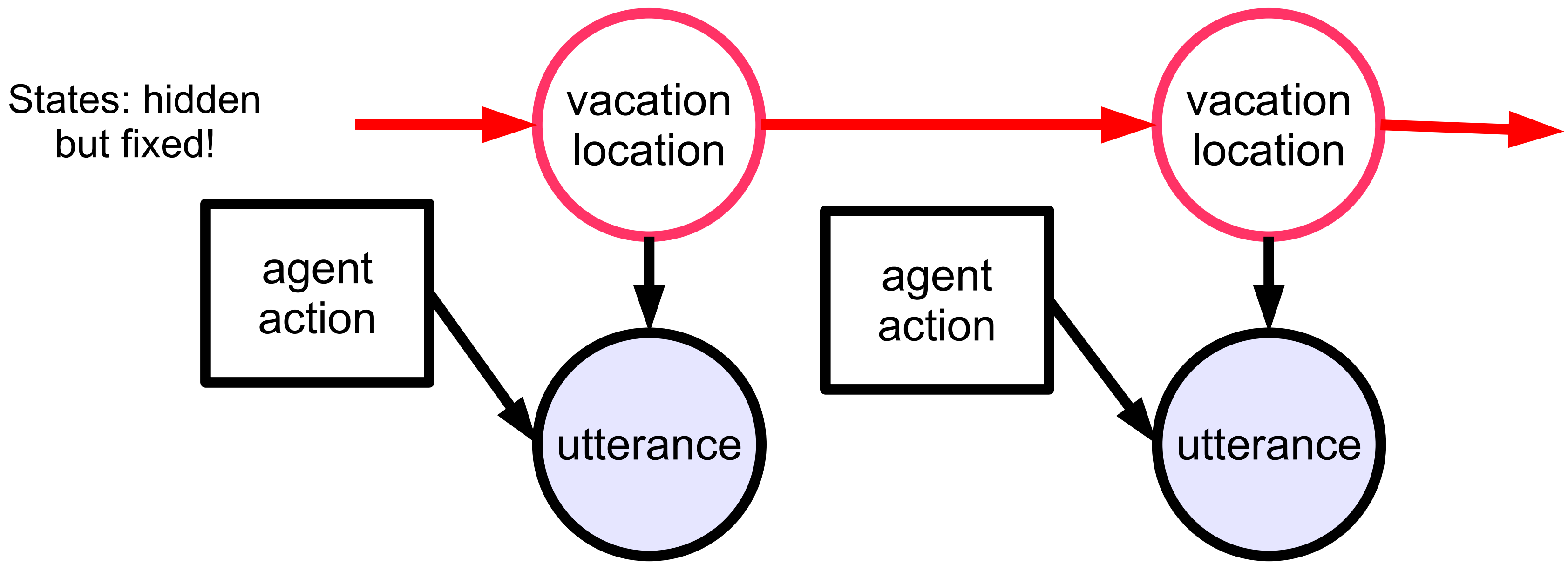


A successful agent must manage **uncertainty** in the dialog.

CSAIL

Toy Example

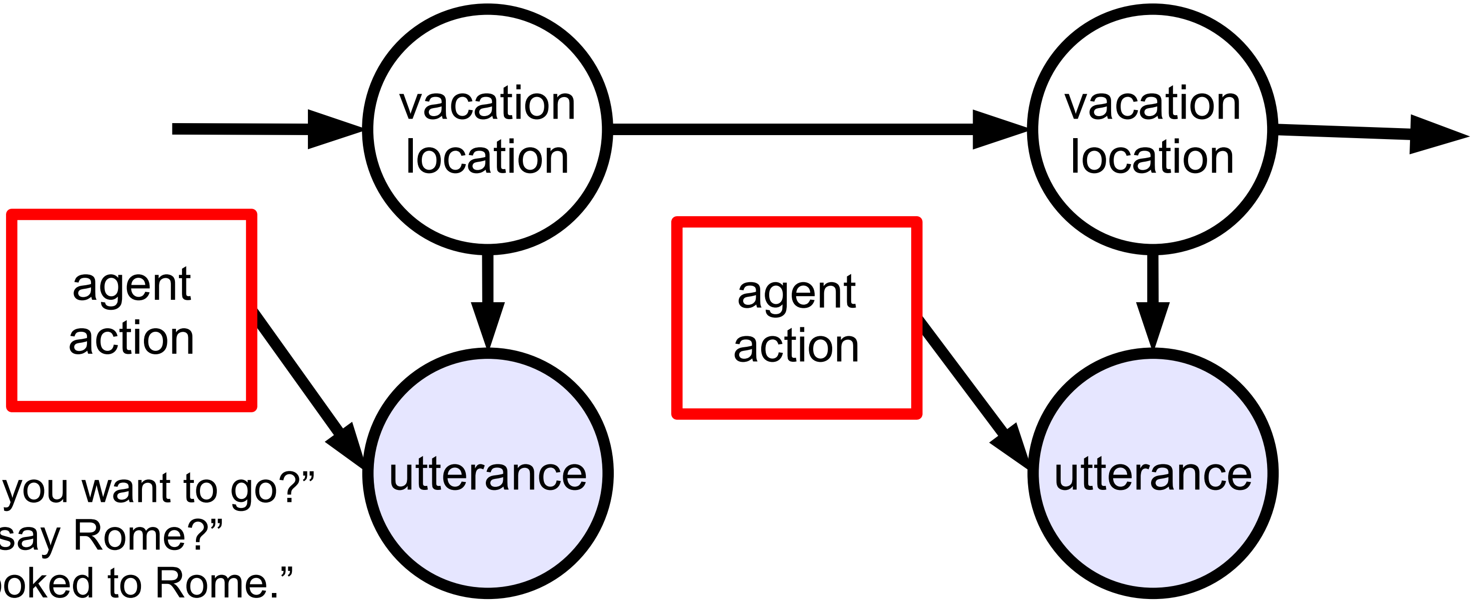
- The **Partially Observable Markov Decision Process** (POMDP) planning framework can optimally manage the dialog uncertainty.
- For our toy example, the model consists of



- Agent rewarded for submitting the correct location and shorter dialogs.

Toy Example

- The **Partially Observable Markov Decision Process** (POMDP) planning framework can optimally manage the dialog uncertainty.
- For our toy example, the model consists of



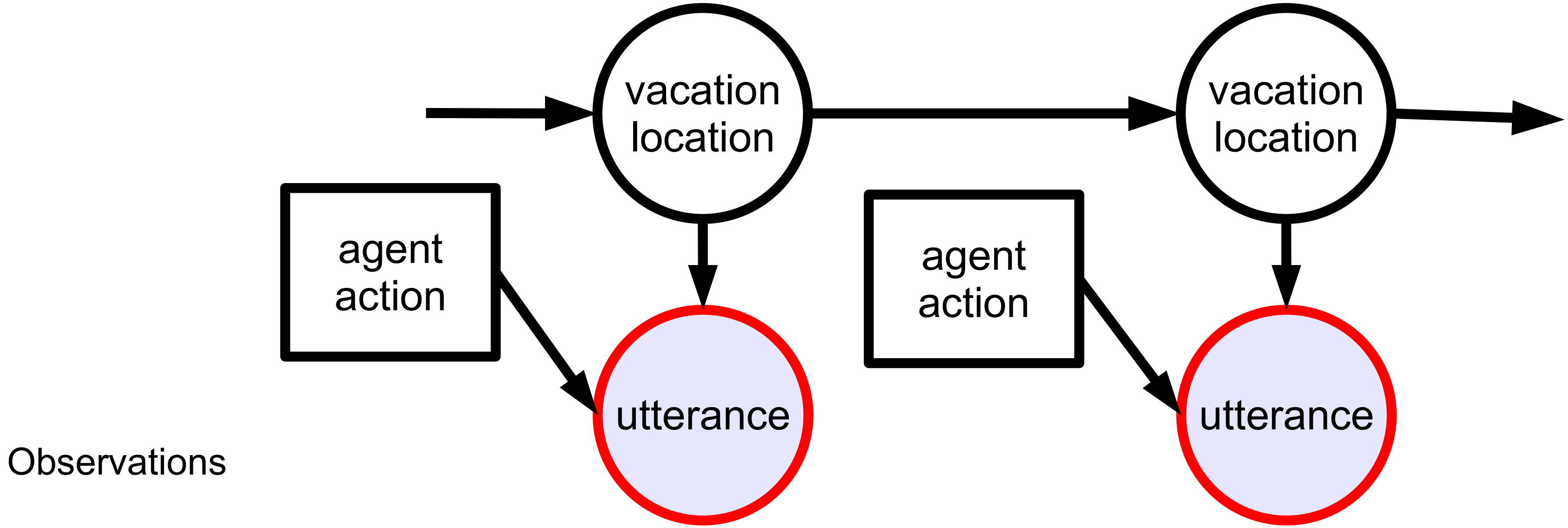
- Actions:
- -Query: “Where do you want to go?”
- -Confirm: “Did you say Rome?”
- -Submit: “You're booked to Rome.”

- Agent rewarded for submitting the correct location and shorter dialogs.

CSAIL

Toy Example

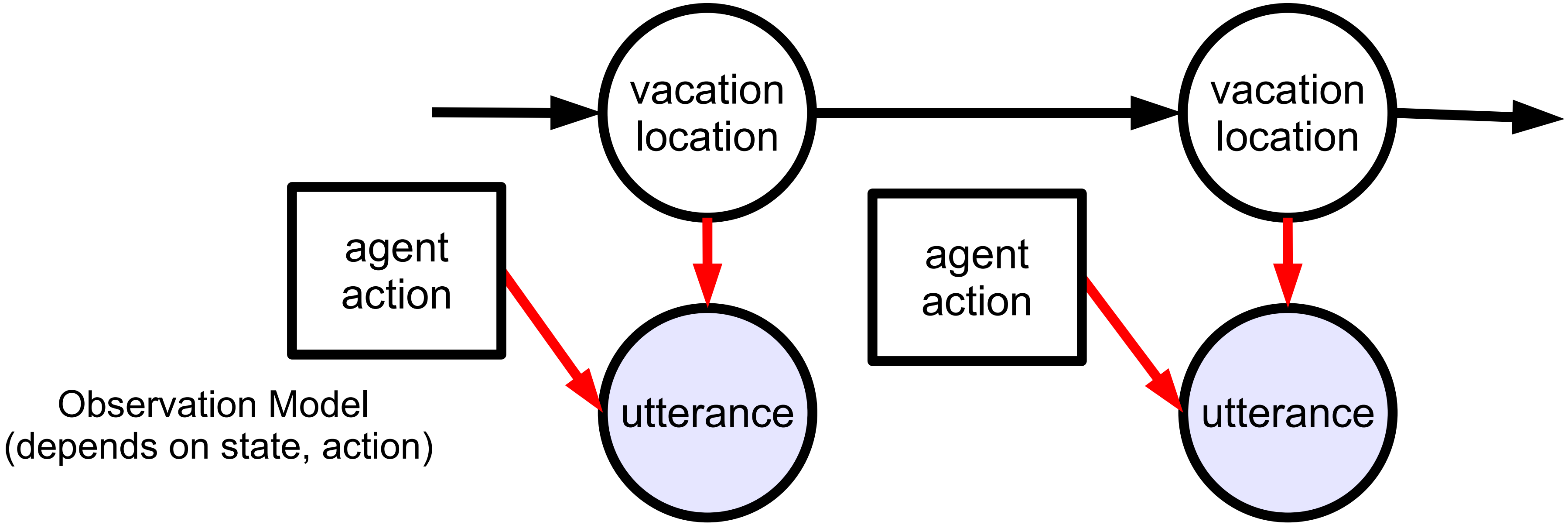
- The **Partially Observable Markov Decision Process** (POMDP) planning framework can optimally manage the dialog uncertainty.
- For our toy example, the model consists of



- Agent rewarded for submitting the correct location and shorter dialogs.

Toy Example

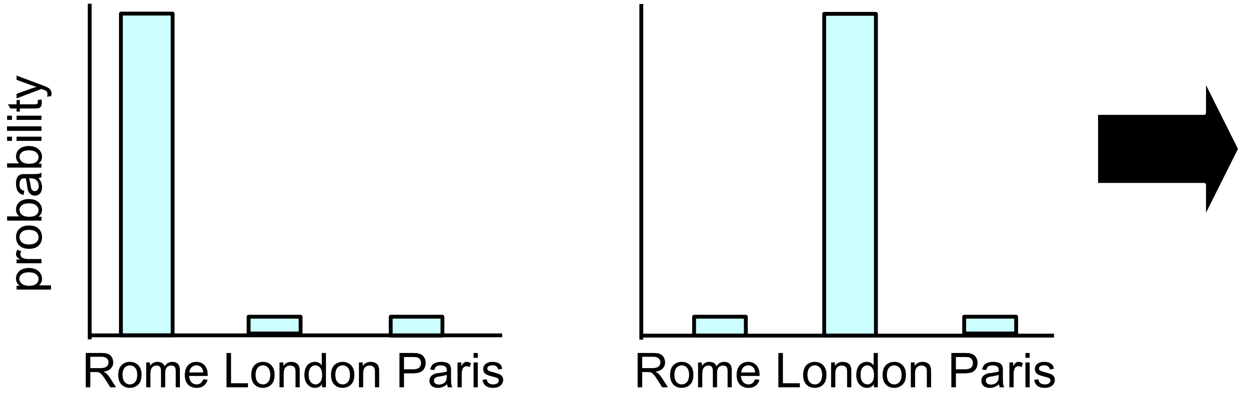
- The **Partially Observable Markov Decision Process** (POMDP) planning framework can optimally manage the dialog uncertainty.
- For our toy example, the model consists of



- Agent rewarded for submitting the correct location and shorter dialogs.

A useful insight

- POMDPs are often hard to solve; the optimal action depends on the agent's belief (probability distribution over the hidden state).
- However, this POMDP has a special structure:

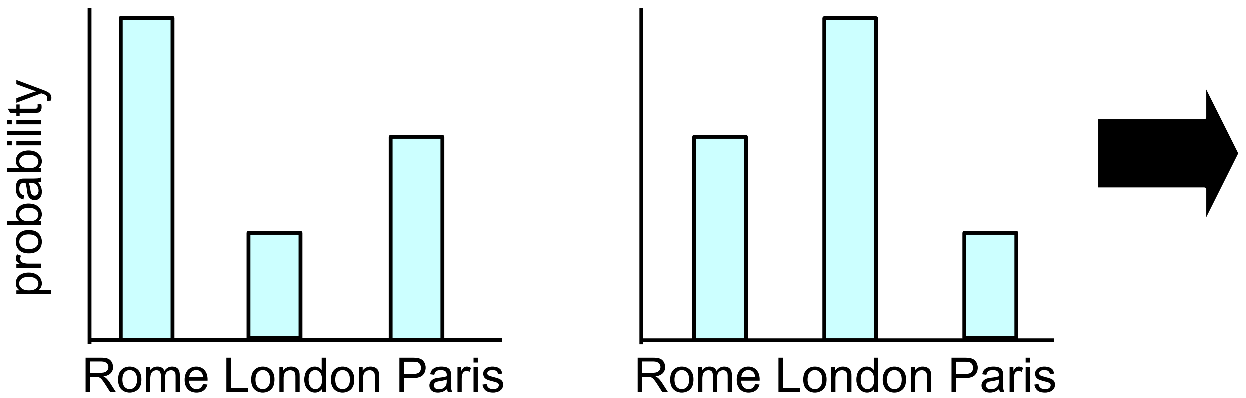


both cases:
best action is
book most
likely state

- Correct type of action depends only on the 'shape' of the belief, not the belief itself.

A useful insight

- POMDPs are often hard to solve; the optimal action depends on the agent's belief (probability distribution over the hidden state).
- However, this POMDP has a special structure:



both cases:
best action is
confirm most
likely state

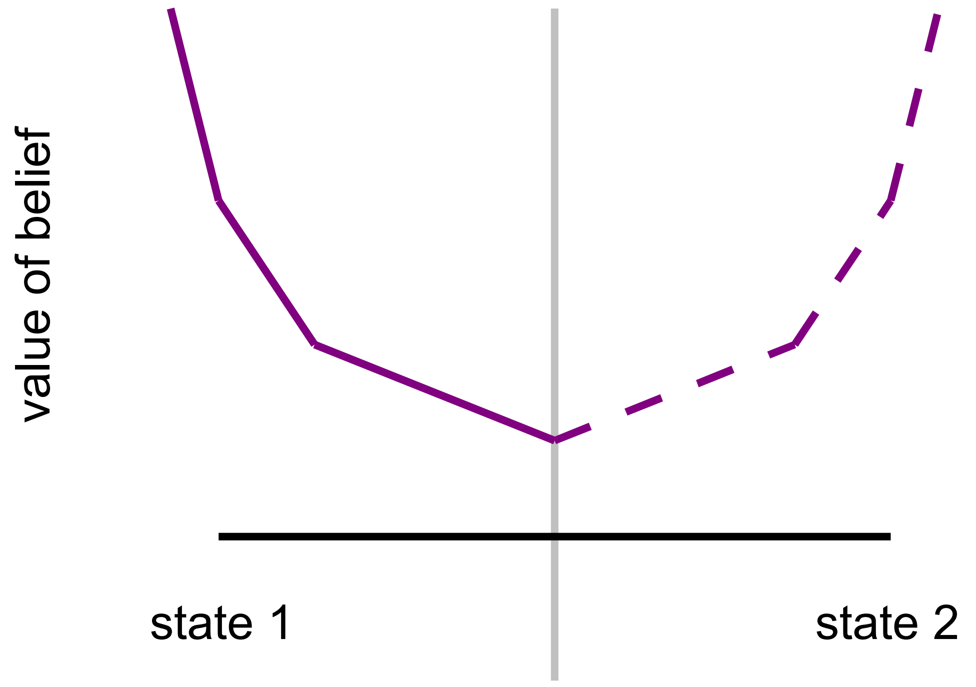
- Correct type of action depends only on the 'shape' of the belief, not the belief itself.

The More General Case

- We can think of many scenarios—booking systems, appointment systems, etc.—that have a similar structure:
  - Goal: determine the identity of a fixed, hidden state.
  - Actions: divide into classes of types with similar but state-dependent effects.
  - Observations: also divide into classes with similar, state-dependent effects.
- In all of these cases, only the shape of the belief matters!
- Note: Other algorithms such as AMDP and Summary POMDP have used this idea, but ours is not an approximation!

Intuitively, why is this useful?

- When solving, we know all permutations of a belief are similar.
- This **exponentially** decreases the belief space we need to consider!



Formalities: sufficient conditions

Theorem: All permutations of the belief will have the same value if,

- for every state permutation $\pi(s)$ and action a
- there exists an action a_π and observation permutation $\pi_{\pi,a}(o)$
- such that
  - $R(s , a) = R(\pi(s) , a_\pi)$
  - $O(o \mid s ,a) = O(\pi_{\pi,a}(o) \mid \pi(s) , a_\pi)$
  - $T(s' \mid s , a) = T(\pi(s') \mid \pi(s) , a_\pi)$

The proof follows from substituting the sufficient condition into the Bellman optimality equations for POMDPs.

Formalities: sufficient conditions

For any action, no matter how we swap around the states,

Theorem: All permutations of the belief v

- for every state permutation $\pi(s)$ and action a
- there exists an action a_π and observation permutation $\pi_{\pi,a}(o)$
- such that
  - $R(s , a) = R(\pi(s) , a_\pi)$
  - $O(o \mid s ,a) = O(\pi_{\pi,a}(o) \mid \pi(s) , a_\pi)$
  - $T(s' \mid s , a) = T(\pi(s') \mid \pi(s) , a_\pi)$

The proof follows from substituting the sufficient condition into the Bellman optimality equations for POMDPs.

Formalities: sufficient conditions

For any action, no matter how we swap around the states,

Theorem: All permutations of the belief v

- for every state permutation $\pi(s)$ and action a
- there exists an action a_π and observation permutation $\pi_{\pi,a}(o)$
- such that
  - $R(s , a) = R(\pi(s) , a_\pi)$
  - $O(o \mid s ,a) = O(\pi_{\pi,a}(o) \mid \pi(s) , a_\pi)$
  - $T(s' \mid s , a) = T(\pi(s') \mid \pi(s) , a_\pi)$

we can find an action and a way to swap around the observations

The proof follows from substituting the sufficient condition into the Bellman optimality equations for POMDPs.

Formalities: sufficient conditions

Theorem: All permutations of the belief v

For any action, no matter how we swap around the states,

- for every state permutation $\pi(s)$ and action a
- there exists an action a_π and observation permutation $\pi_{\pi,a}(o)$
- such that
  - $R(s , a) = R(\pi(s) , a_\pi)$
  - $O(o \mid s ,a) = O(\pi_{\pi,a}(o) \mid \pi(s) , a_\pi)$
  - $T(s' \mid s , a) = T(\pi(s') \mid \pi(s) , a_\pi)$

we can find an action and a way to swap around the observations

The proof follows from substituting the s
Bellman optimality equations for POM

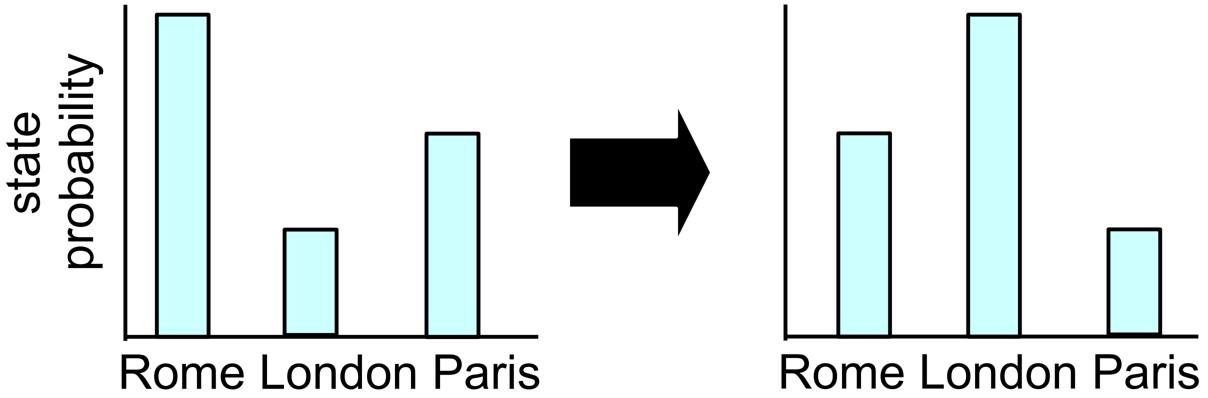
so that the parameters are remain the same

CSAIL

Formalities: sufficient conditions

Example: given

- $\pi(s)$, π(Rome) = London
- Action "Confirm Rome"



Let a_π = "Confirm London" and observation permutation $\pi_{\pi,a}(o) = o$

- R(Rome , "Confirm Rome") = R(π(Rome) , "Confirm London")
- O(Yes | Rome , "Confirm Rome") = O(Yes | London , "Confirm London")
- T(Rome | Rome , "Confirm Rome") = T(London | London , "Confirm London")

Practicalities: how do we use this?

We'll illustrate the approach for a simple point-based POMDP solution technique, but the idea—which **exponentially** reduces the size of the belief space—can be applied to more sophisticated POMDP solvers.

Practicalities: how do we use this?

Generic point-based POMDP solution scheme:

- Sample a set of beliefs.
- Loop:
  - Compute action-observation value vectors:

$$\Gamma^{a,o} = \{\alpha \mid \alpha(s) = \gamma \sum_{s' \in S} T(s' \mid s, a) O(o \mid s', a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Compute the action value vectors:

$$\Gamma_b^a = R(., a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = argmax_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$

CSAIL

Practicalities: how do we use this?

Adaptation to the permutable POMDP:

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

$$\Gamma^{a,o} = \{\alpha \mid \alpha(s) = \gamma \sum_{s' \in S} T(s' \mid s, a) O(o \mid s', a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma_b^a = R(.,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = argmax_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$

CSAIL

Practicalities: how do we use this?

Adaptation to the permutable POMDP:

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

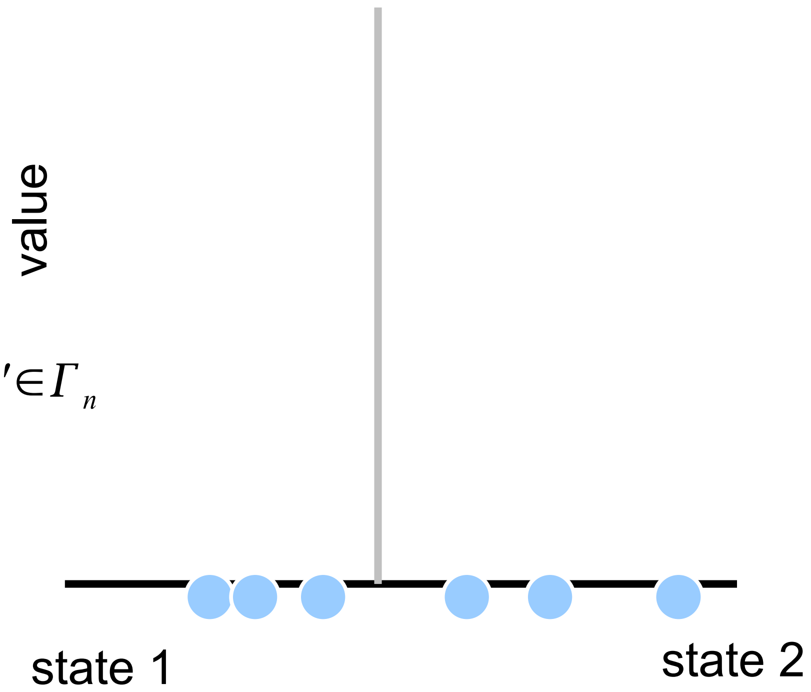
$$\Gamma^{a,o} = \{\alpha | \alpha(s) = \gamma \sum_{s' \in S} T(s'|s,a) O(o|s',a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma^a_b = R(.,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = argmax_{\Gamma^a_b, a} (\Gamma^a_b \cdot b)$$



Practicalities: how do we use this?

Adaptation to the permutable POMDP:

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

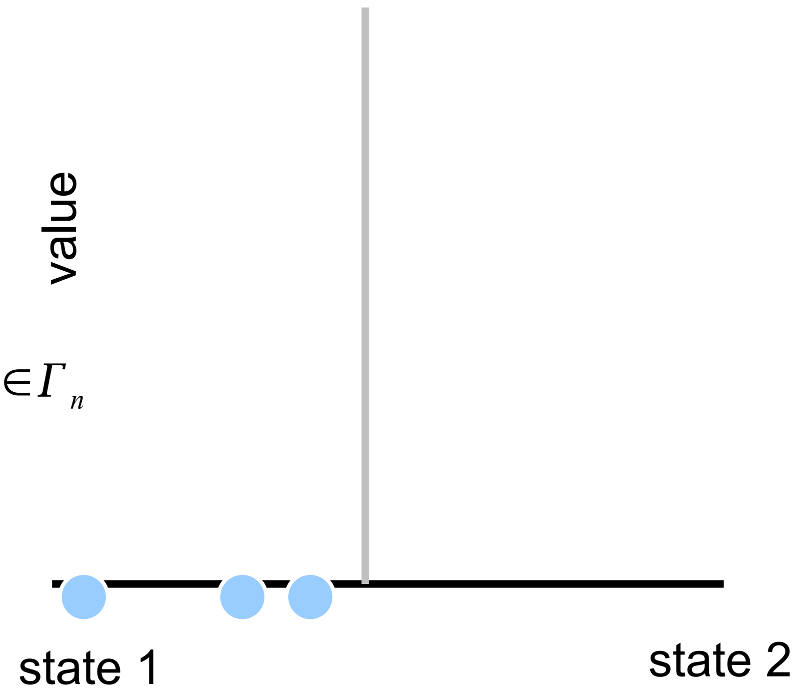
$$\Gamma^{a,o} = \{\alpha | \alpha(s) = \gamma \sum_{s' \in S} T(s'|s,a) O(o|s',a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma_b^a = R(.,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = argmax_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$



Practicalities: how do we use this?

Adaptation to the permutable POMDP:

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

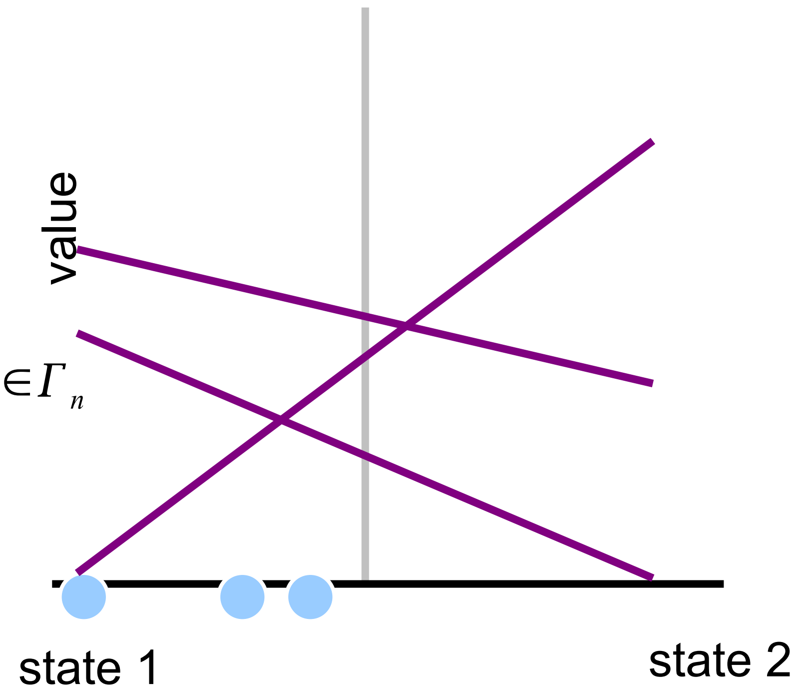
$$\Gamma^{a,o} = \{\alpha \mid \alpha(s) = \gamma \sum_{s' \in S} T(s' \mid s, a) O(o \mid s', a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma_b^a = R(.,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = argmax_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$



Practicalities: how do we use this?

Adaptation to the permutable POMDP:

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

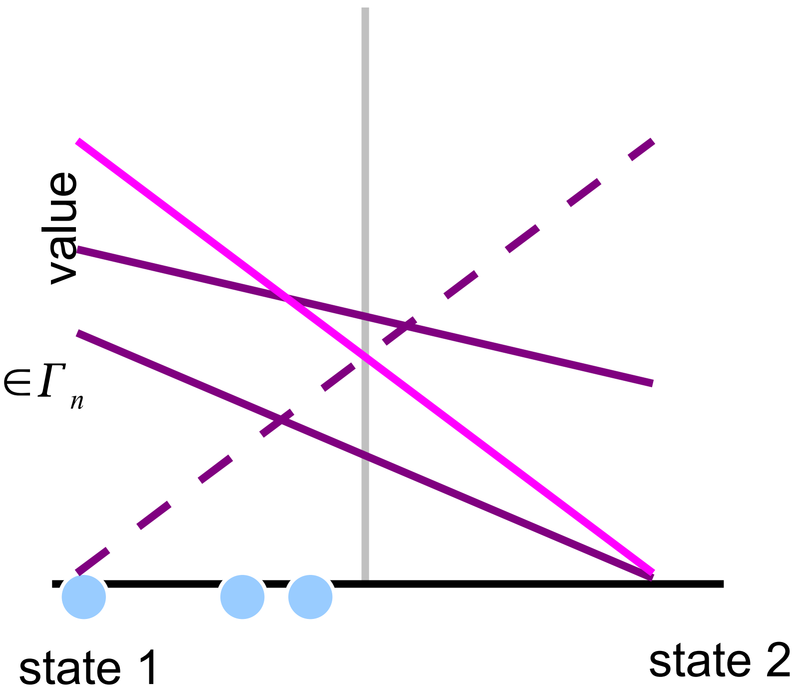
$$\Gamma^{a,o} = \{\alpha \mid \alpha(s) = \gamma \sum_{s' \in S} T(s' \mid s, a) O(o \mid s', a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma_b^a = R(.,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = argmax_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$



Practicalities: how do we use this?

Adaptation to the permutable POMDP:

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

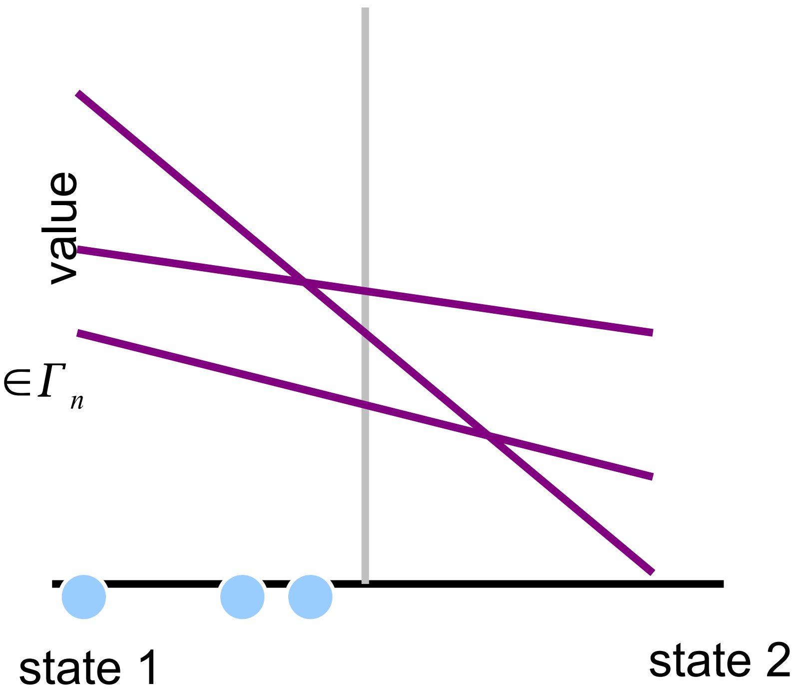
$$\Gamma^{a,o} = \{\alpha \mid \alpha(s) = \gamma \sum_{s' \in S} T(s' \mid s, a) O(o \mid s', a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma_b^a = R(., a) + \sum_{o \in O} \operatorname{argmax}_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = \operatorname{argmax}_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$



Practicalities: how do we use this?

Adaptation to the permutable POMDP:

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

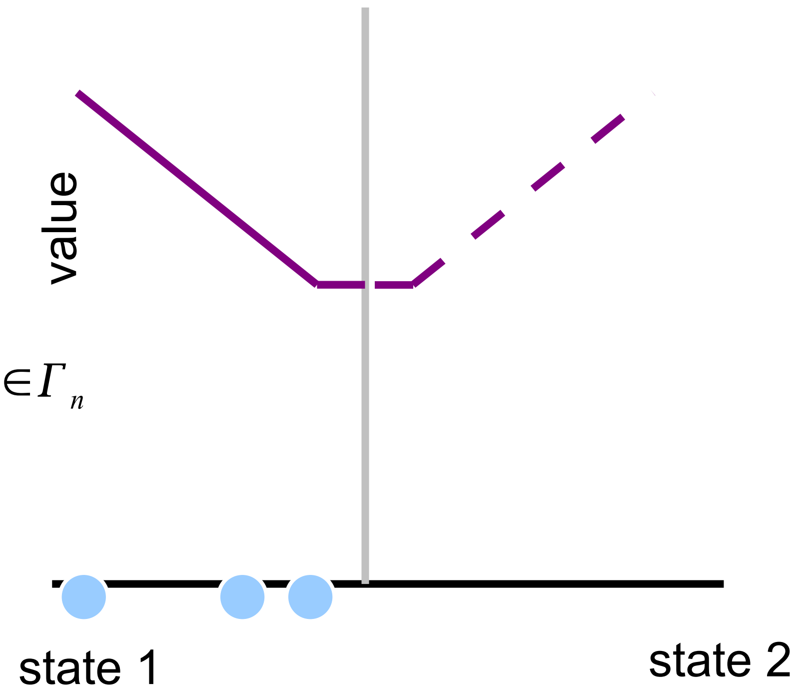
$$\Gamma^{a,o} = \{\alpha \mid \alpha(s) = \gamma \sum_{s' \in S} T(s' \mid s, a) O(o \mid s', a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma_b^a = R(.,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

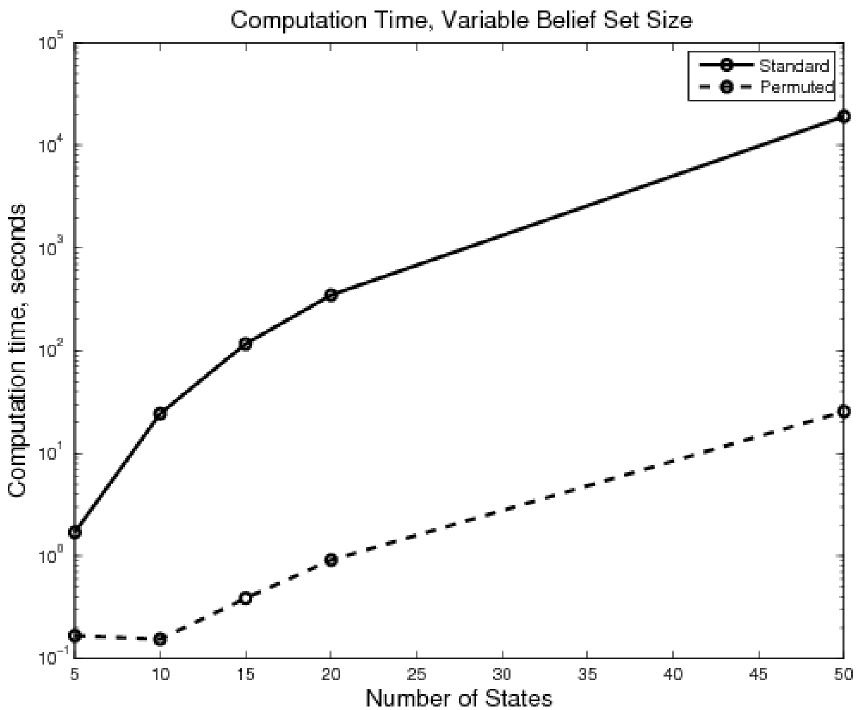
$$\Gamma_{n+1} = argmax_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$

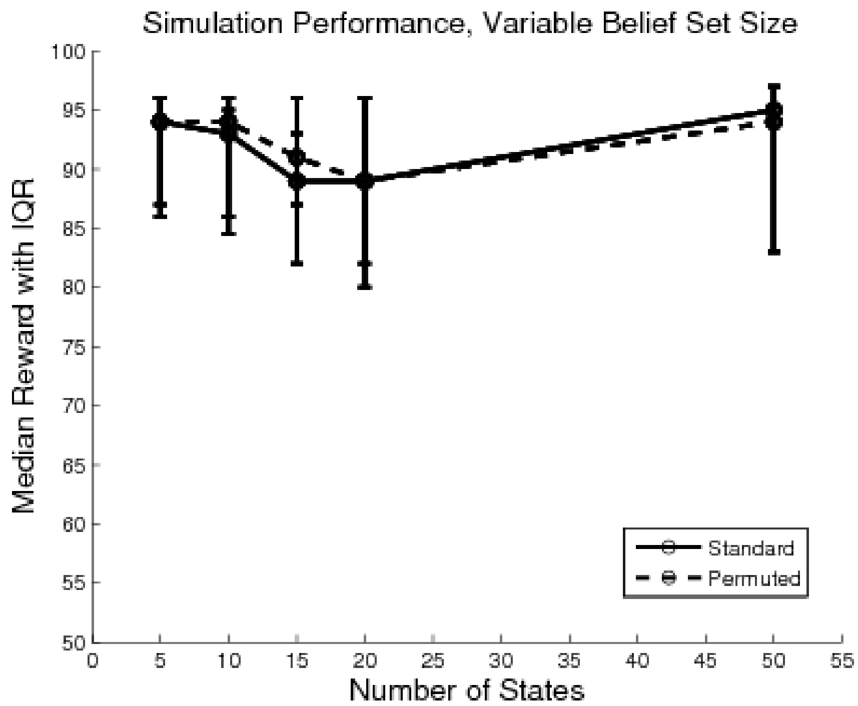


Results

- Scenario:
  - Three types of actions: “ask,” “confirm,” “submit”
  - Unique observation associated with each state
  - Pr[hear correct state] from “ask” = .5
  - Pr[hear correct confirmation] from “confirm” = .8
  - Varied the number of states.
- Two types of tests:
  - let the belief set size grow with the number of states (more fair to the generic algorithm)
  - fix belief set size (more reasonable if time or memory is limited)

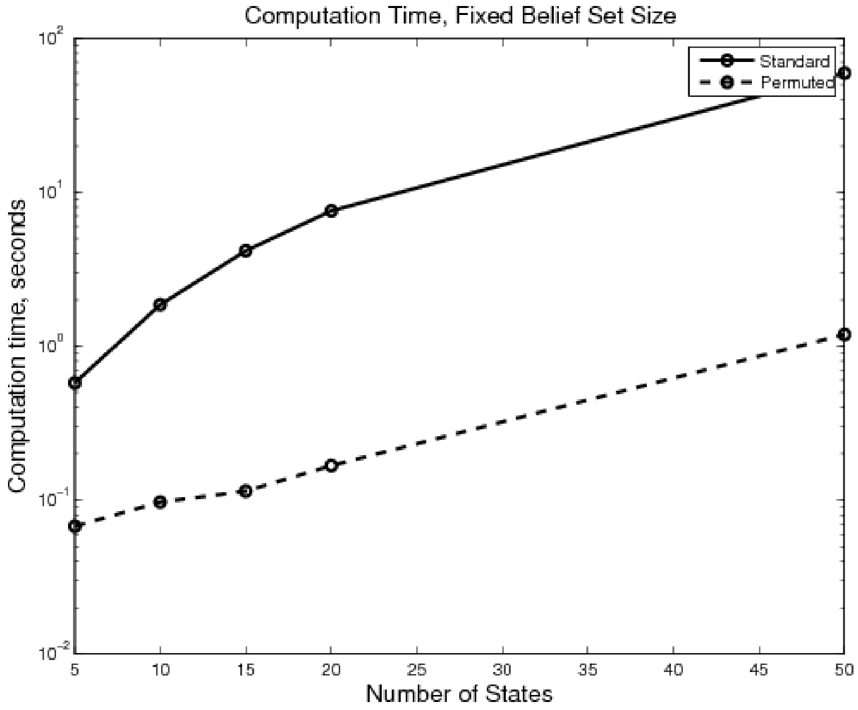
Growing belief set size

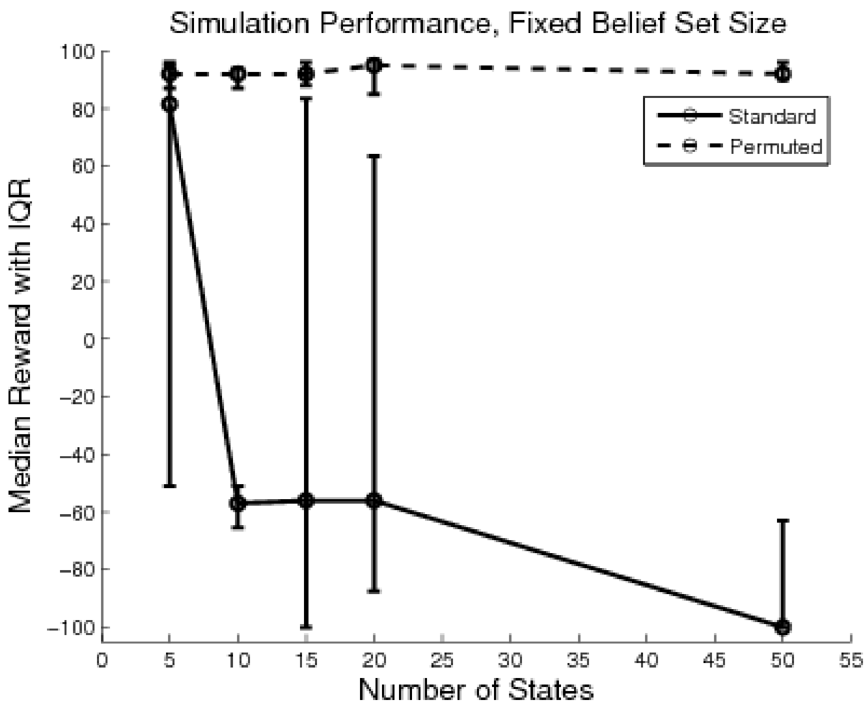




CSAIL

Fixed belief set size





Conclusion

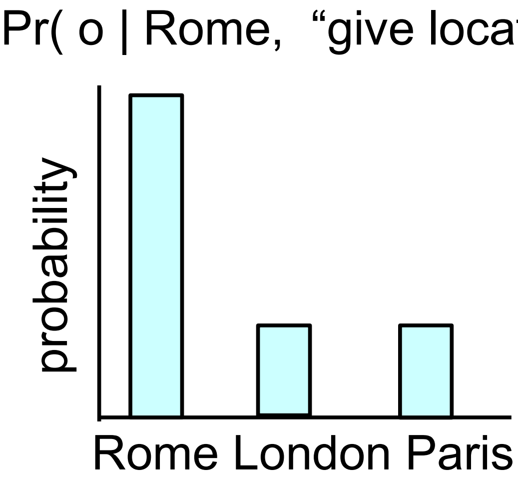
- Useful trick for solving POMDPs with a particular structure
  - can also be used if a substructure is permutable.
  - useful in learning situations in which a policy must be evaluated whenever parameters change.
- Future work:
  - determine a more general set of necessary conditions.
  - are there approximate ways to apply this trick for POMDPs with not quite permutable structure?

CSAIL

Thank-you

Formalities: sufficient conditions

Example: given π(s) and action "Give location"



Let a_π = "Give location" and $\pi_{\pi,a}(o_s) = o_{\pi(s)}$:

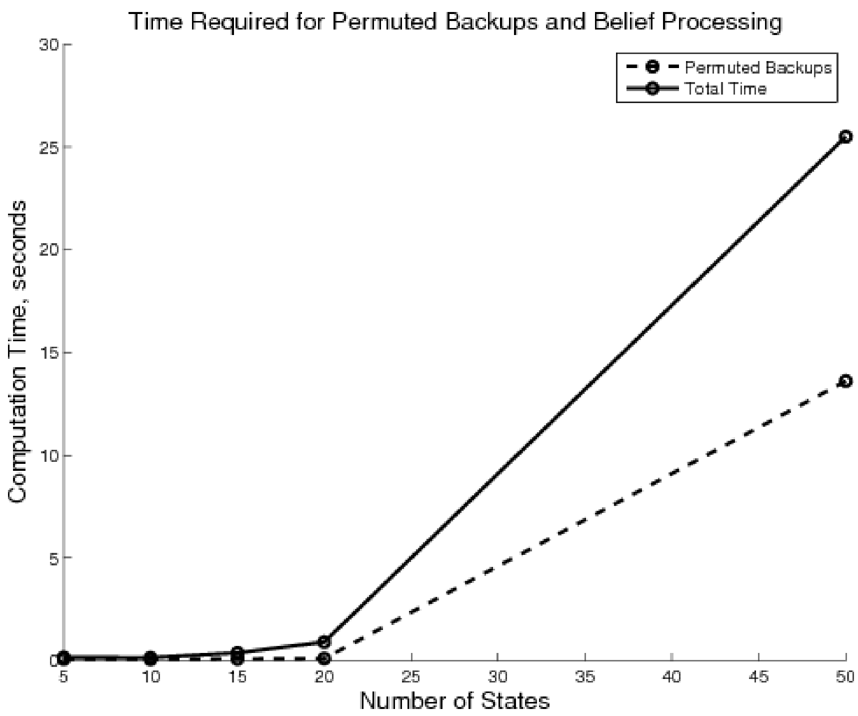
- R(Rome , "Give location") = R(π(Rome) , "Give location")
- O(Rome | Rome , "Give location") = O(London | London , "Give location")
- T(Rome | Rome , "Give location") = T(London | London , "Give location")

C S A I L

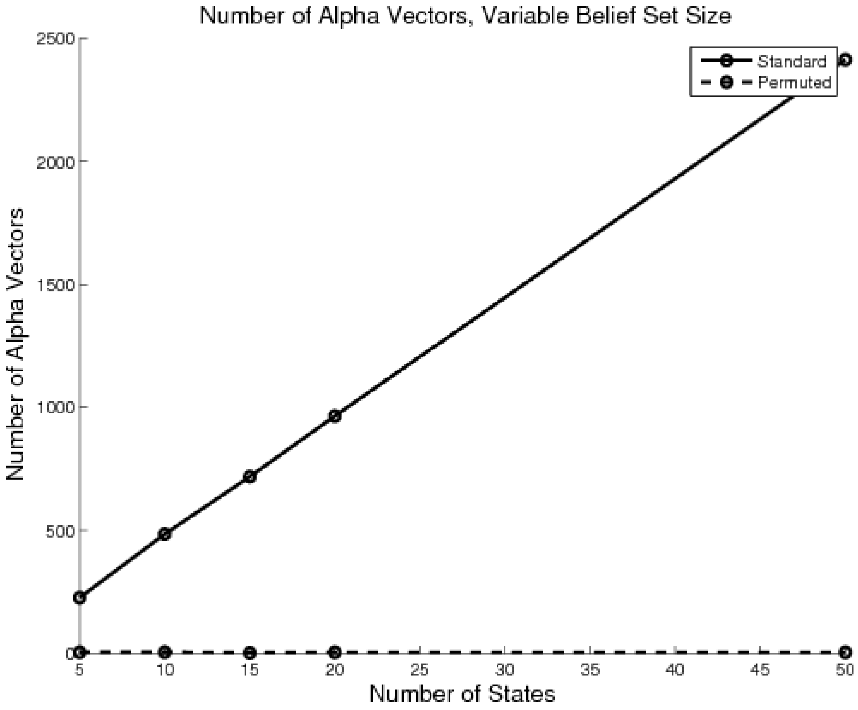
Application References

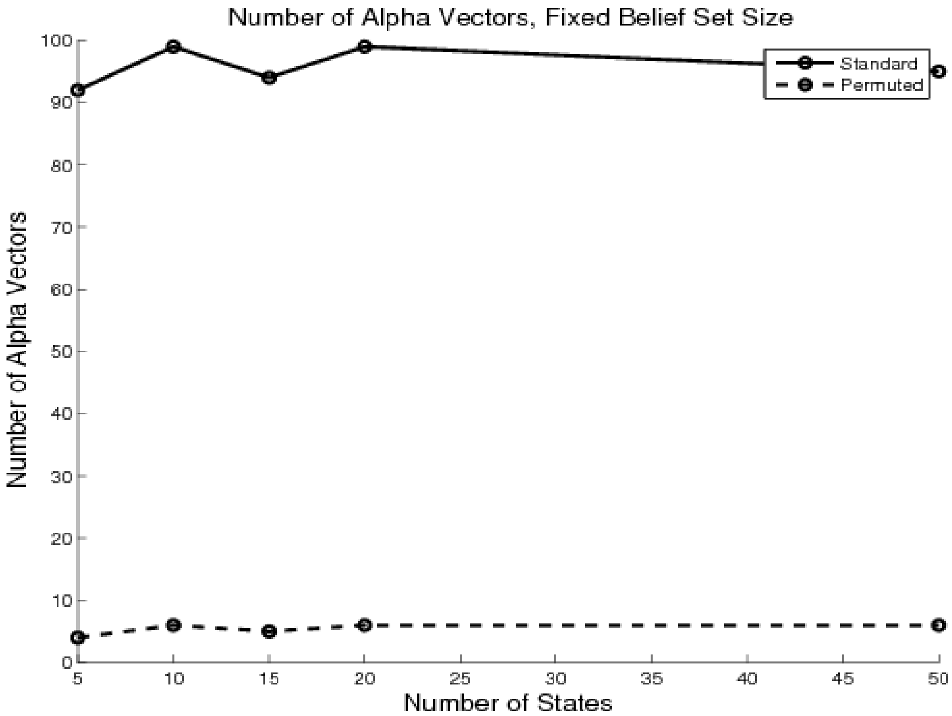
- Williams and Young, Scaling up POMDPs for dialog management: the summary POMDP method, IEEE ASRU Workshop 2005.
- Roy, Pineau, and Thrun, Spoken dialog management using probabilistic reasoning, Proceedings of the 38th Meeting of the ACL, 2000.
- Regan, Cohen, and Poupart, The advisor POMDP: a principled approach to trust through reputation in electronic markets. Conference on Privacy, Security, and Trust, 2005.
- Doshi and Roy, Efficient model learning for dialog management, Human-Robot Interaction, 2007.
- Boutilier, A POMDP formulation of preference elicitation problems, Proceedings of the 18th National Conference on Artificial Intelligence, 2002.

Time Spent on Backups



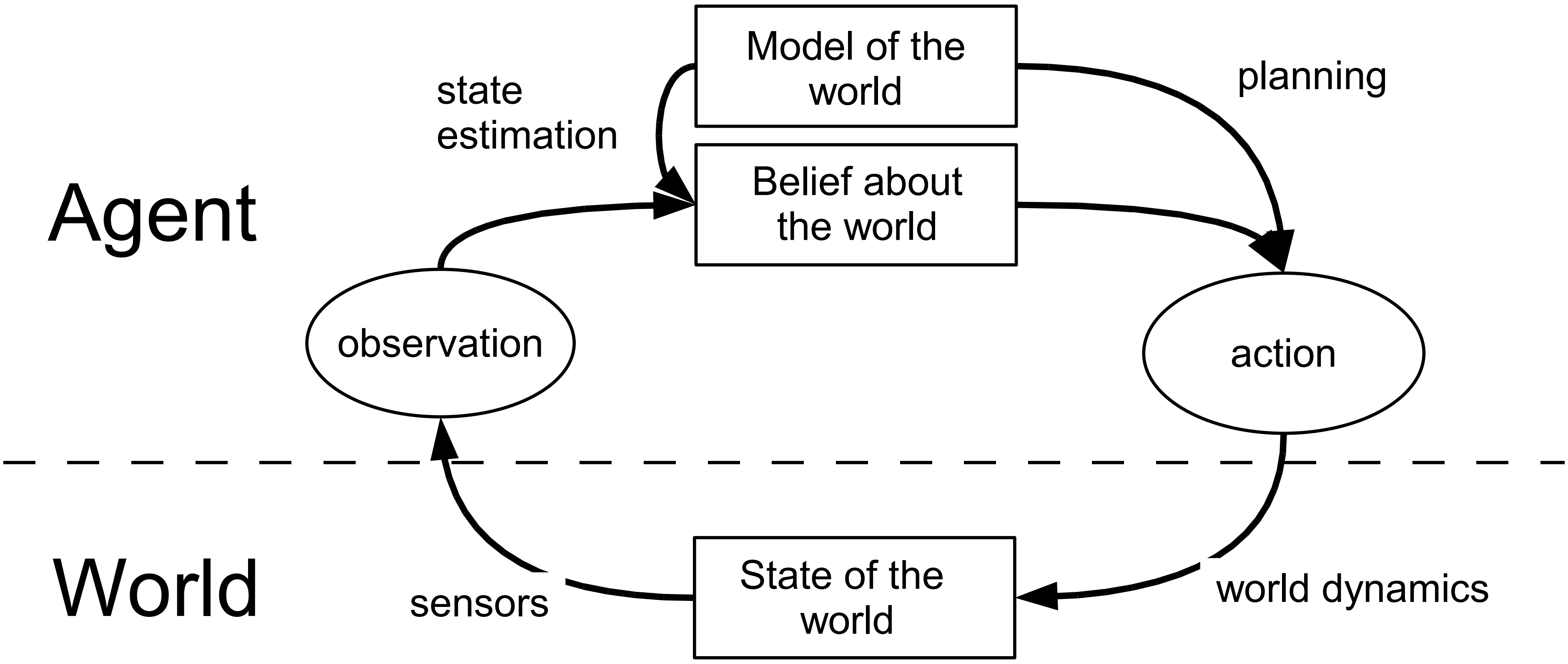
Alpha Set Sizes





CSAIL

Interaction Model

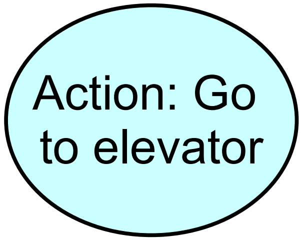


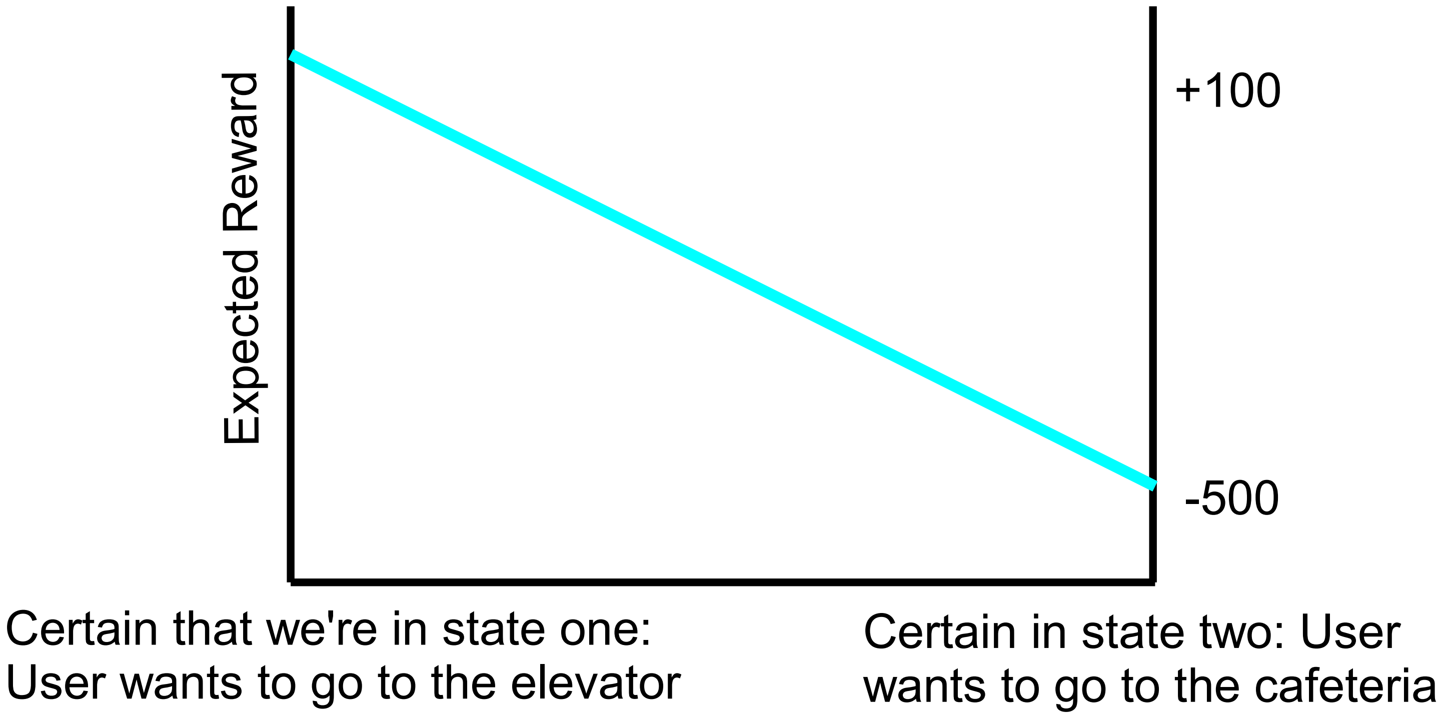
Solving a POMDP

$$
\begin{aligned}
V(b) &= \max_{a \in A} Q(b,a), \\
Q(b,a) &= R(b,a) + \gamma \sum_{b' \in B} T(b'|b,a) V(b'), \\
Q(b,a) &= R(b,a) + \gamma \sum_{o \in O} \Omega(o|b,a) V(b_a^o),
\end{aligned}
$$

Dialog Model: Solving the POMDP

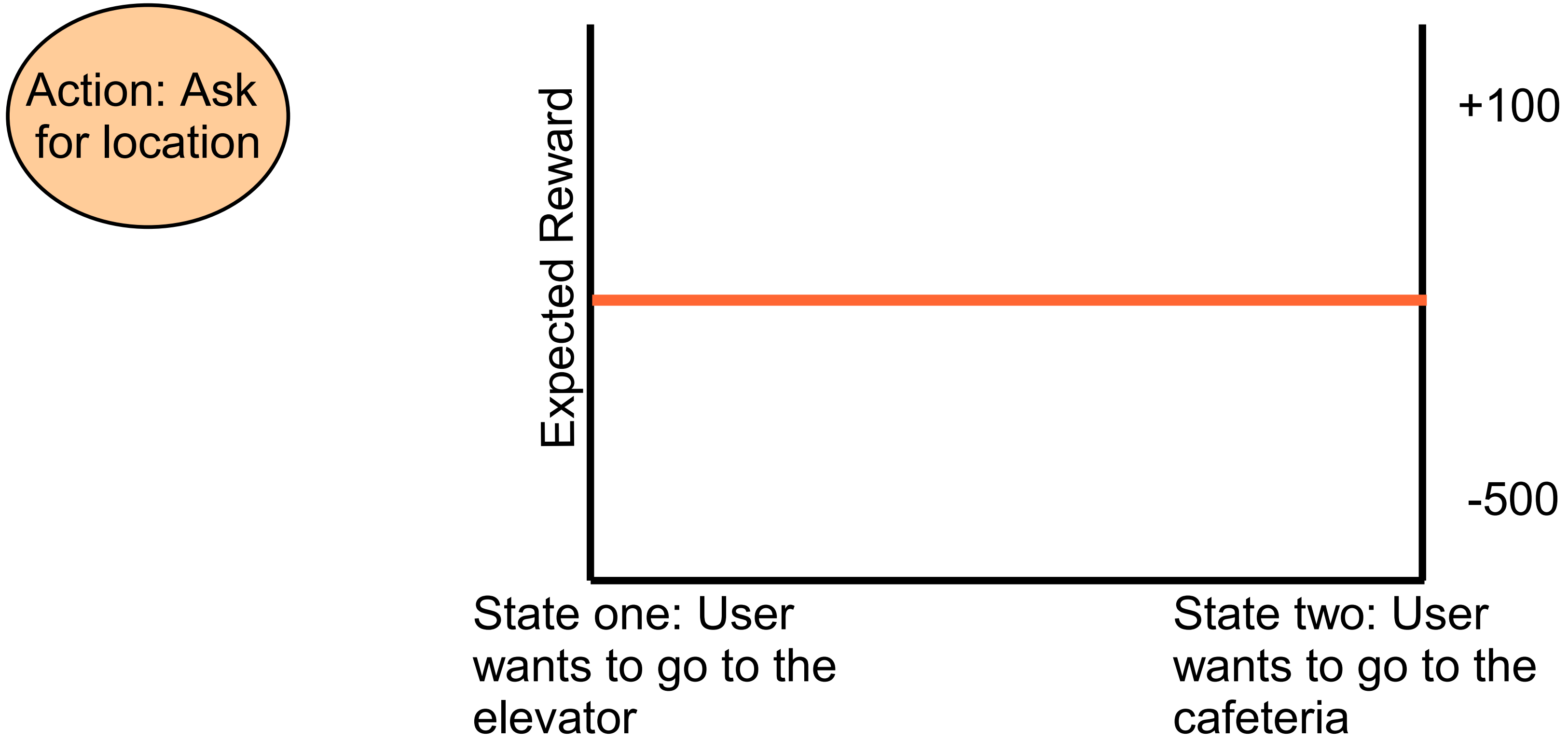
We think of the previous recursions as building a policy tree...





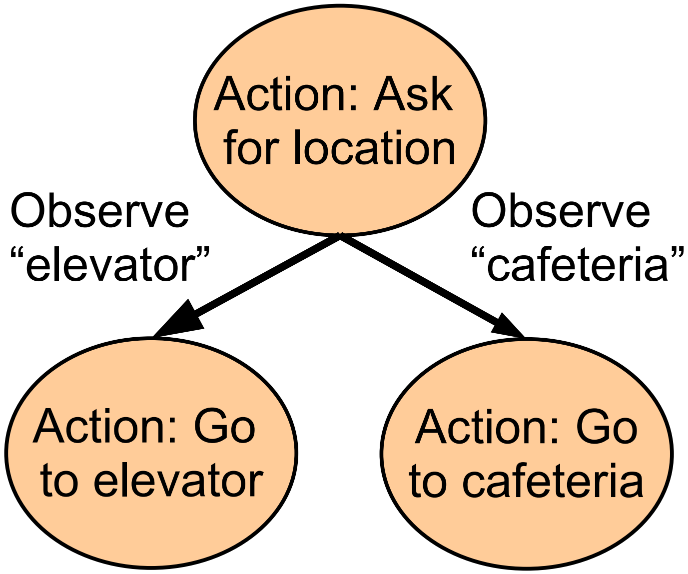
Dialog Model: Solving the POMDP

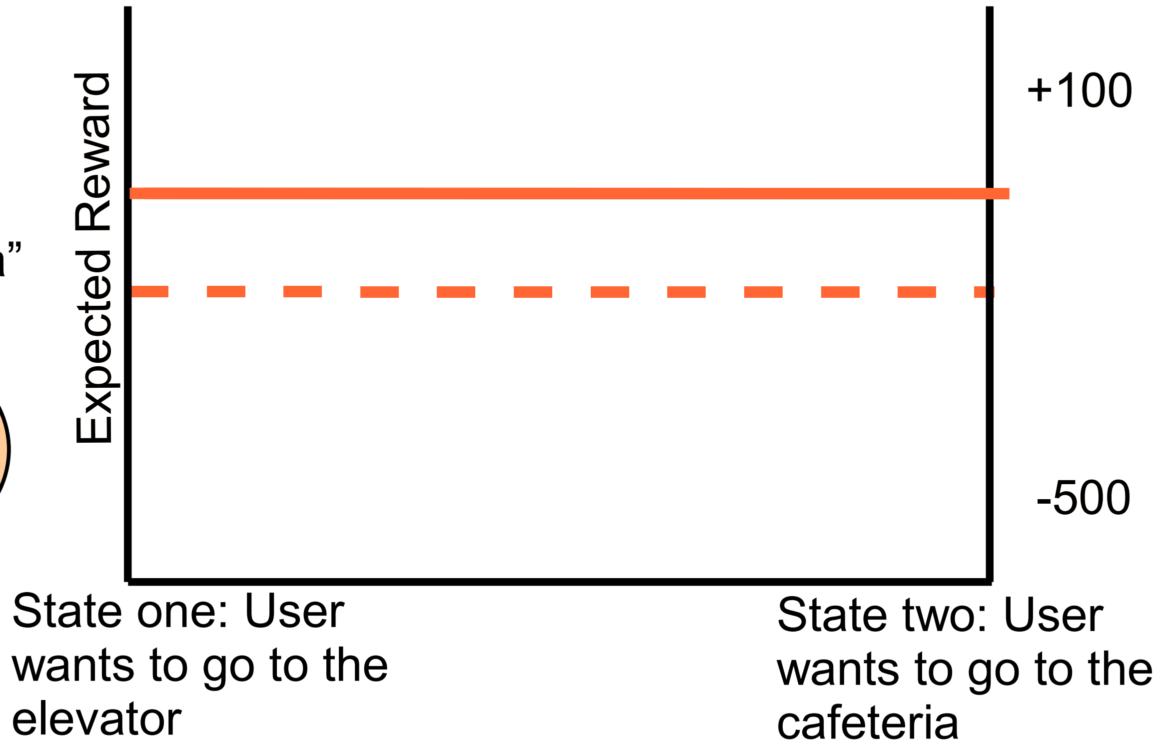
We think of the previous recursions as building a policy tree...



Dialog Model: Solving the POMDP

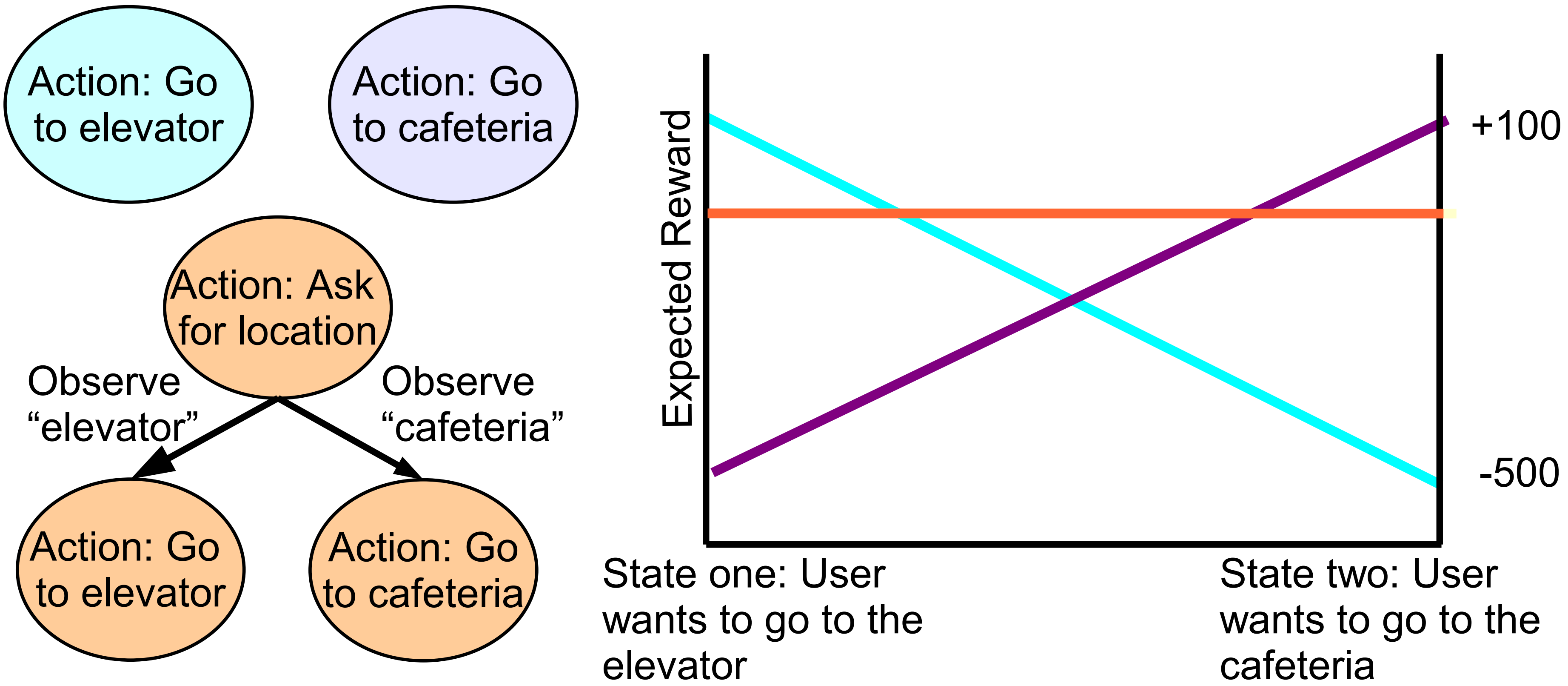
We think of the previous recursions as building a policy tree; planning ahead increases our expected reward.





Dialog Model: Solving the POMDP

Given multiple trees, we can determine the most appropriate action:



Practicalities: how do we use this?

- We'll illustrate the approach for a simple point-based POMDP solution technique, but the idea—which effectively reduces the size of the belief space—can be applied to more sophisticated POMDP solvers.
- Begin with some notation:
  - Γ: value function; dot(Γ , b) is the value of being in belief b.
  - Γ^a_b: action value funciton; dot(Γ^a_b , b) is the value of being in b and taking action a.
  - $\Gamma^{a,o}$: action-observation value function; dot($\Gamma^{a,o}$, b) is the value of being in b, taking action a, and seeing observation o.

Practicalities: how do we use this?

Adaptation to the permutable POMDP:

Get a canonical set of beliefs

- Sample a set of beliefs.
- Sort beliefs in descending order and remove nearby points.
- Loop:
  - Compute action-observation value vectors:

$$\Gamma^{a,o} = \{\alpha \mid \alpha(s) = \gamma \sum_{s' \in S} T(s' \mid s, a) O(o \mid s', a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$$

  - Sort the $\Gamma^{a,o}$ in descending order.
  - Compute the action value vectors:

$$\Gamma_b^a = R(,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$$

  - Compute the new value function:

$$\Gamma_{n+1} = argmax_{\Gamma_b^a, a} (\Gamma_b^a \cdot b)$$

CSAIL

Practicalities: how do we use this?

Why we sort the $\Gamma^{a,o}$ vectors:

- Previous step: $\Gamma^{a,o} = \{\alpha | \alpha(s) = \gamma \sum_{s' \in S} T(s'|s,a) O(o|s',a) \alpha'(s')\}, \forall \alpha' \in \Gamma_n$
- Next step: $\Gamma_b^a = R(,a) + \sum_{o \in O} argmax_{\alpha \in \Gamma^{a,o}} (\alpha \cdot b)$