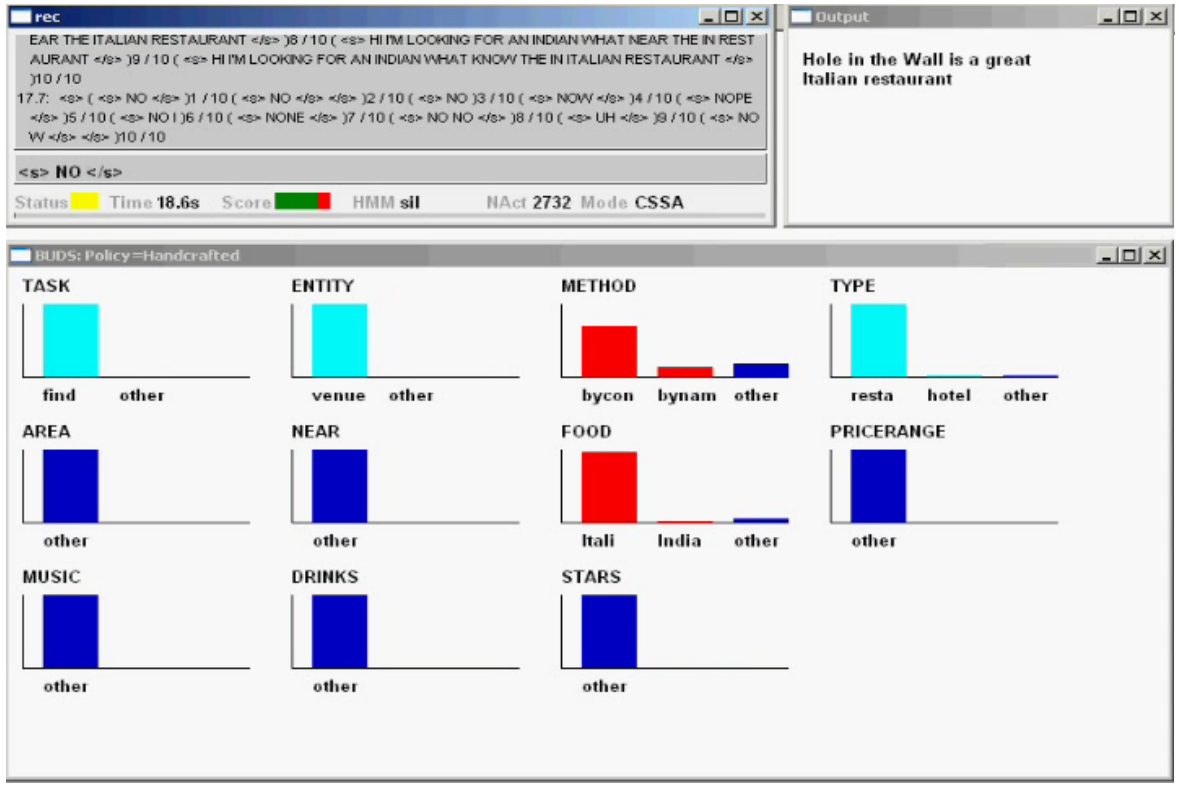
Partially Observable Markov Decision Processes

Lee Wee Sun
School of Computing
leews@comp.nus.edu.sg

Dialog Systems



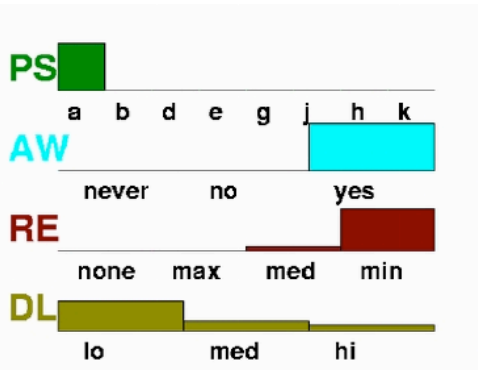
- Aim: Find out what the person wants
- Actions: Ask appropriate questions
- Observations: Output of speech recognizer

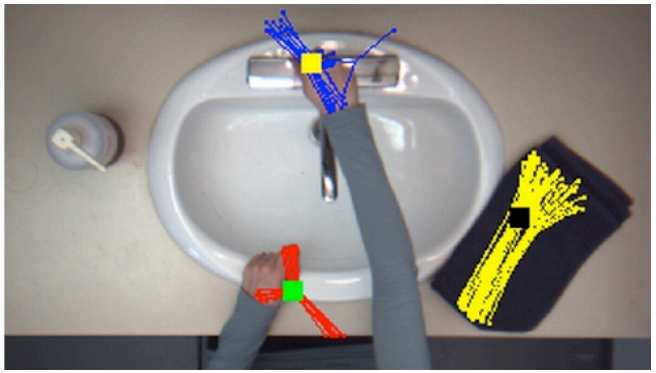
Video from http://www.research.att.com/people/Williams_Jason_D/

POMDP

Williams & Young, 2007

Assistive Technology





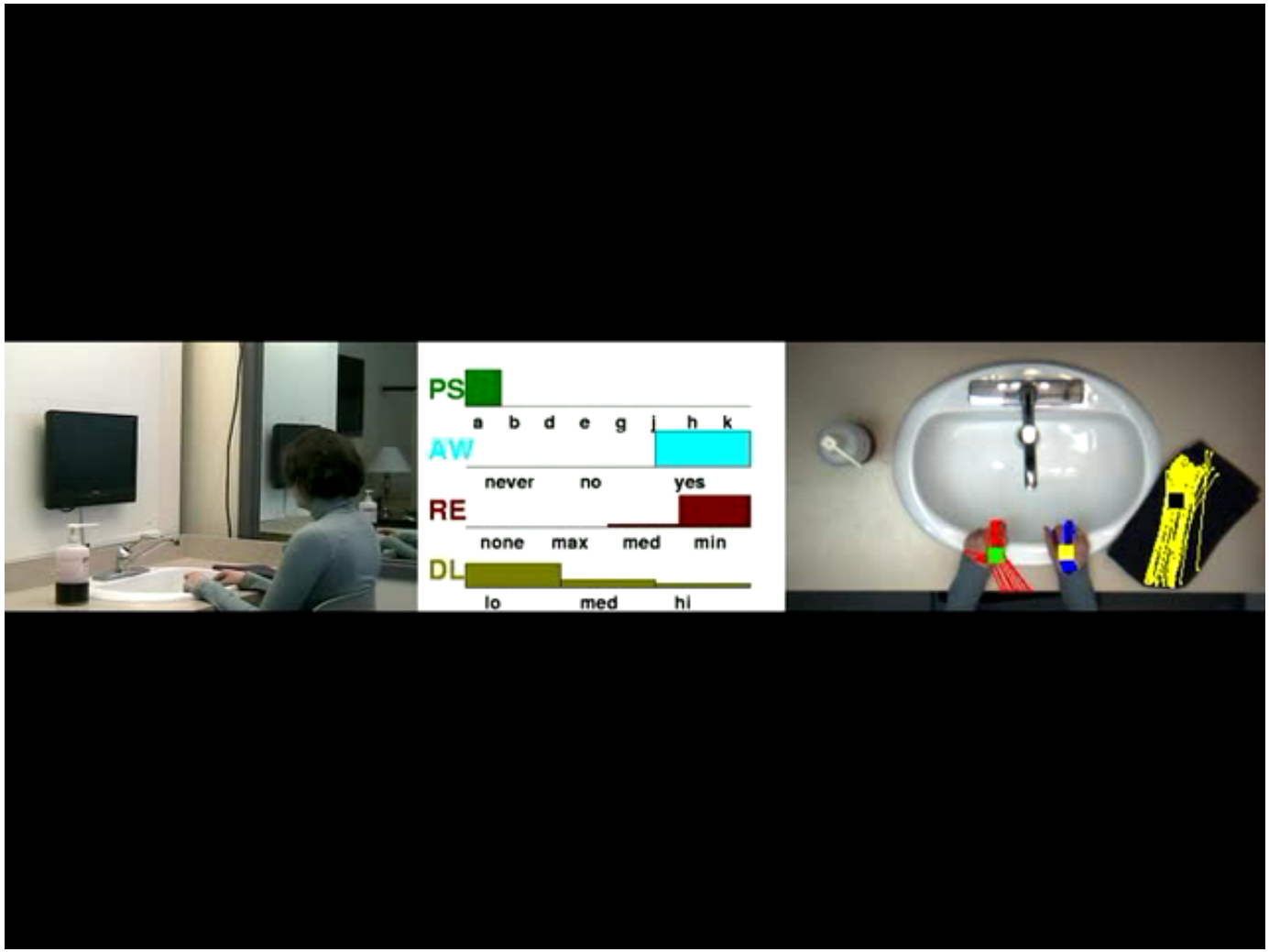
- Aim: Assist person with dementia in handwashing
- Actions: Prompt the person with suggestions when appropriate
- Observations: Video of activity

Video from http://www.cs.uwaterloo.ca/~jhoey/research/coach/index.php

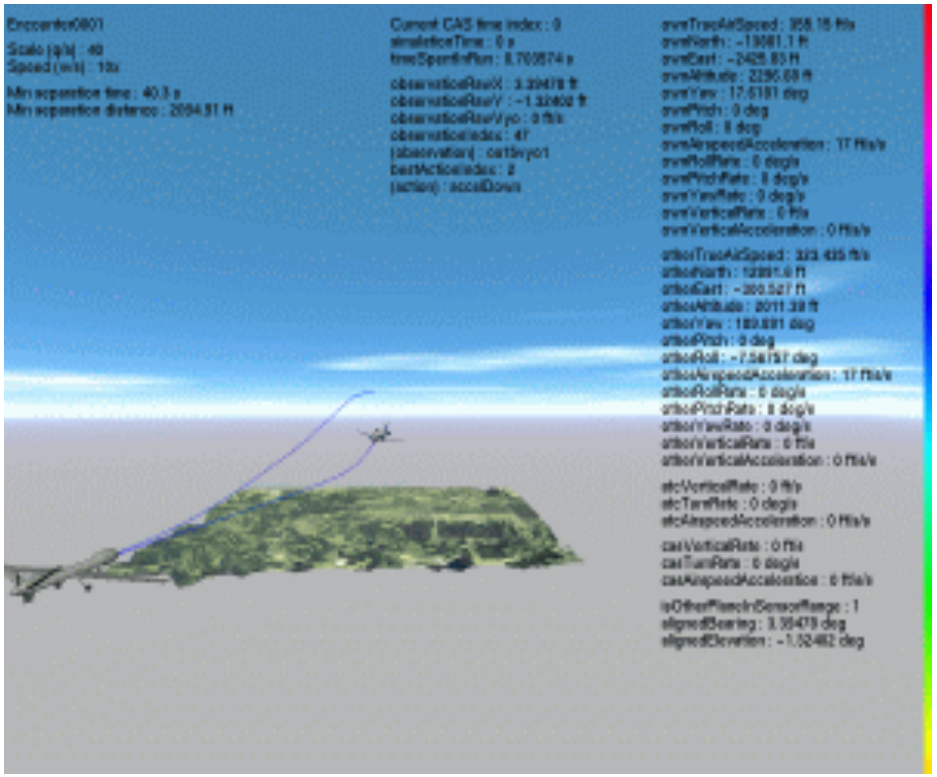
POMDP

Hoey, Von Bertoldi, Poupart, Mihailidis 2007

Assistive Technology



Aircraft Collision Avoidance System

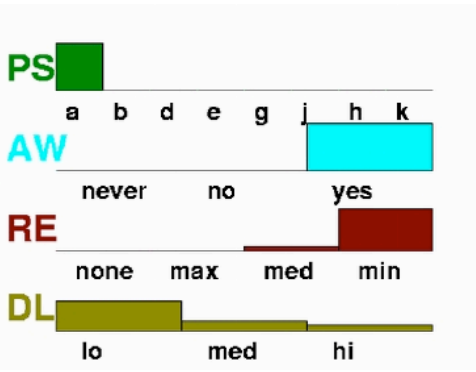


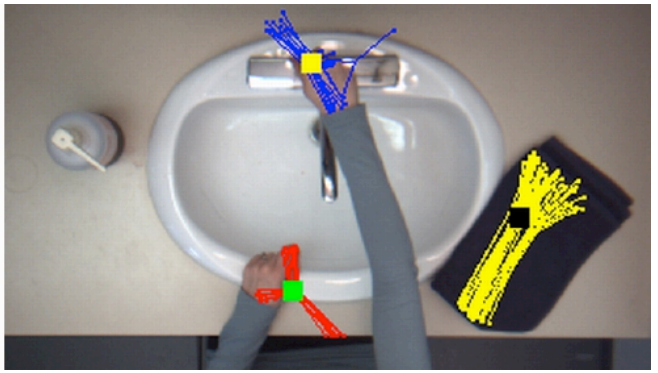
- Aim: Avoid collision with nearby aircraft
- Actions: Maneuver the UAV
- Observations: Limited view sensors

Image from http://web.mit.edu/temizer/www/selim/

Temitzer, Kochenderfer, Kaelbling, Lozano-Perez, Kuchar 2010
Bai, Hsu, Lee, Kochenderfer 2011

Bayesian Reinforcement Learning





- Aim: Assist person with dementia in handwashing without knowing some parameters of model
- Actions: Prompt the person with suggestions to guide the person and at the same time learn the model
- Observations: Video of activity

Poupart, Vlassis, Hoey, Regan 2006

POMDP

NUS National University of Singapore

- Commonalities in the examples
  - Need to learn, estimate or track the current state of the system from history of actions and observations
  - Based on current state estimate, select an action that leads to good outcomes, not just currently but also in future (planning)

Outline

- Overview
- Definitions
- Basic properties
- Intractability and Easier Subclasses
- Large state spaces
- Online search

Powerful but Intractable

- Partially Observable Markov Decision Process (POMDP) is a very powerful modeling tool
- But with great power
  … comes great intractability!

No known way to solve it quickly

No small policy

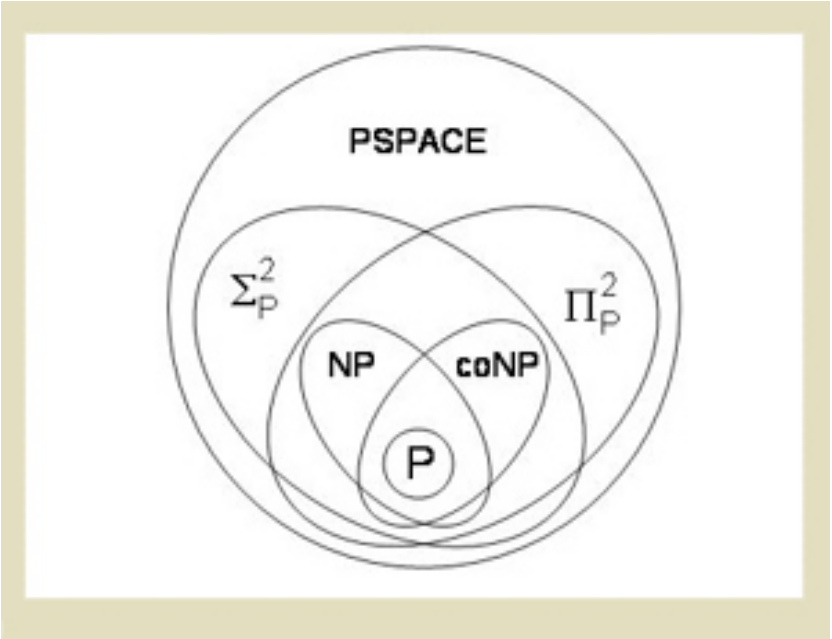


Image from http://ocw.mit.edu/courses/mathematics/18-405j-advanced-complexity-theory-fall-2001/

NUS National University of Singapore

- Philosophy: Okay if cannot solve the really hard problems, but want to be able to solve easy problems masquerading as hard ones …

What are the easier sub-classes?

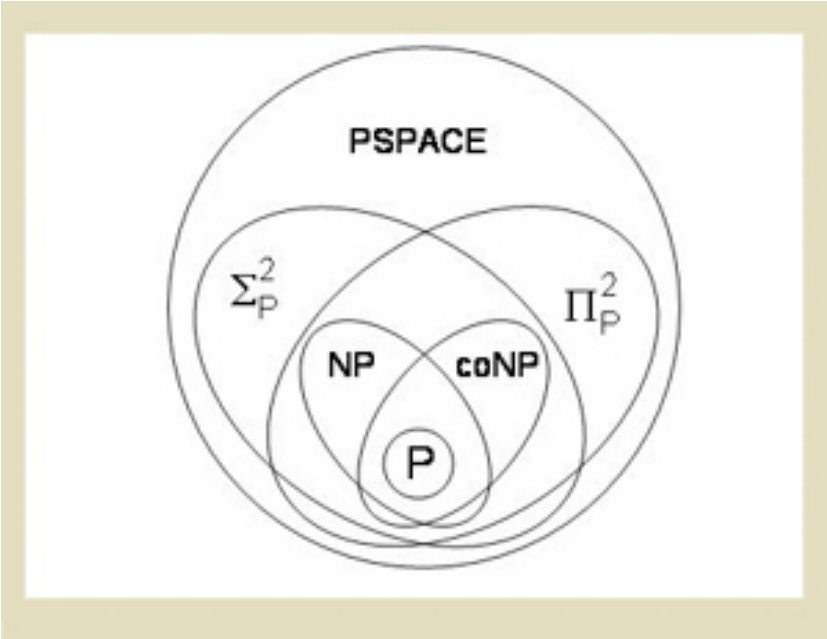
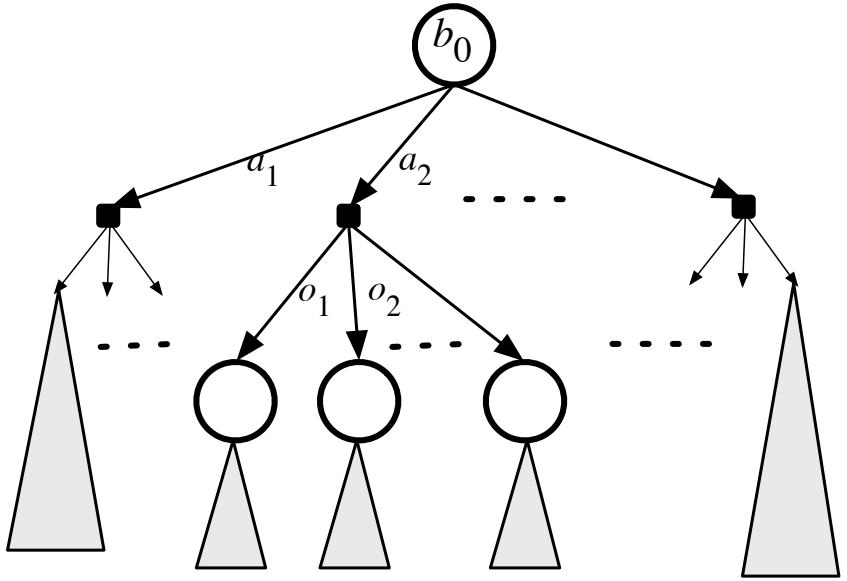


Image from http://ocw.mit.edu/courses/mathematics/18-405j-advanced-complexity-theory-fall-2001/

Belief Space Size

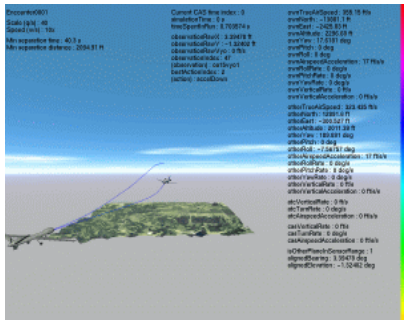
- For POMDPs, we work with beliefs: probability distribution of states
- Assume known initial belief b_0
- Solve only for space reachable from b_0

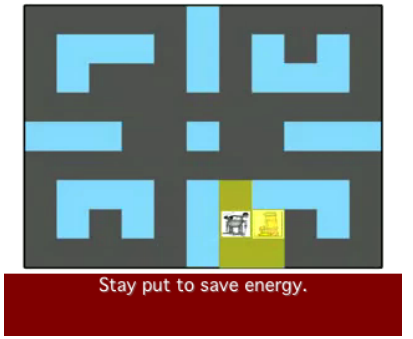


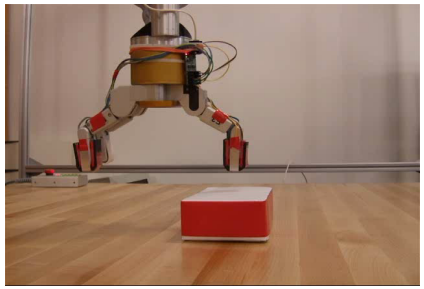
  - Disregarding the difficulty of approximate belief evaluation …
    - If reachable belief space is small, efficient approximation possible
    - If belief reachable when acting optimally is small, small policy exists

Characterizing Belief Space

- Belief space size can grow exponentially with the number of states
- But state space size often poor indicator of belief space size, e.g. can be exponentially smaller
  - When some state variables are observed
  - When belief can be factored







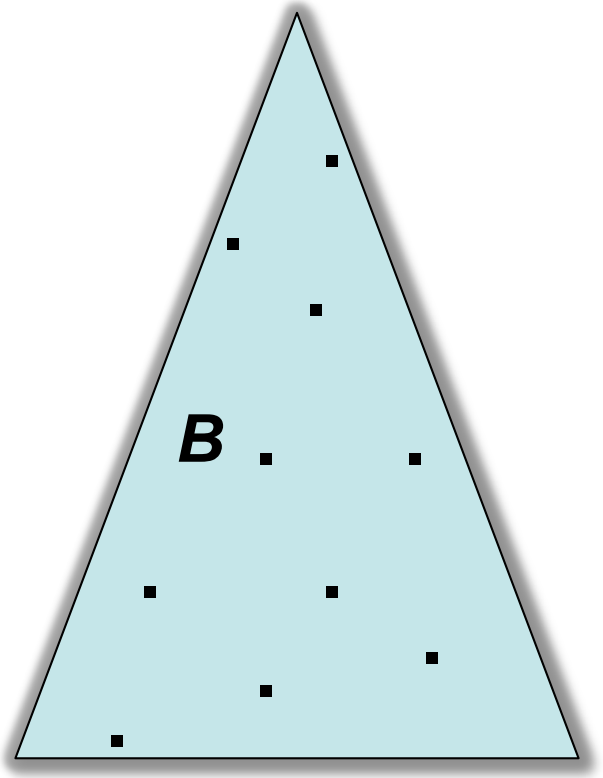
Robot observed in collision avoidance, tracking, grasping

Slot belief factored in slot filling dialog system

Point Based Algorithms

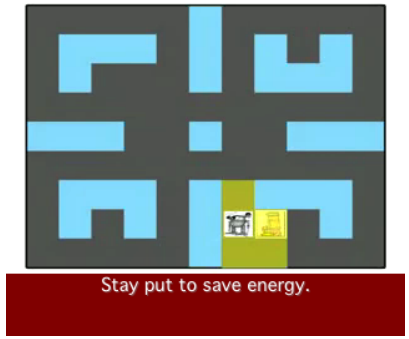
Approximates belief space with a finite set of belief

- Properly designed,
  - Able to exploit small reachable belief for efficient computation
  - Able to exploit heuristics and branch and bound algorithms to get small policy when space reachable under optimal actions is small



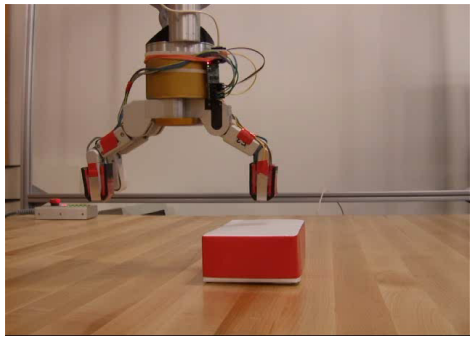
State Space Size

- Moderate state space size



tracking

navigation



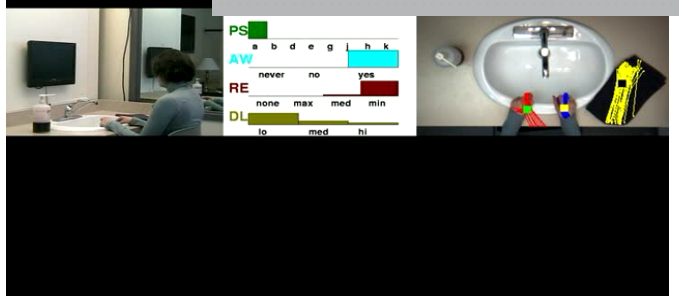
simple grasping

- Exact belief and policy evaluation

- Very large, continuous state spaces

dialog

collision avoidance



assistance

- Approximate belief and policy evaluation

Very Large State Spaces

- Techniques still essentially the same when we think a small policy exists
- But use either
  - Symbolic representation for belief and value function, or
  - Approximate belief and policy evaluation
    - Monte Carlo methods

Online Algorithms

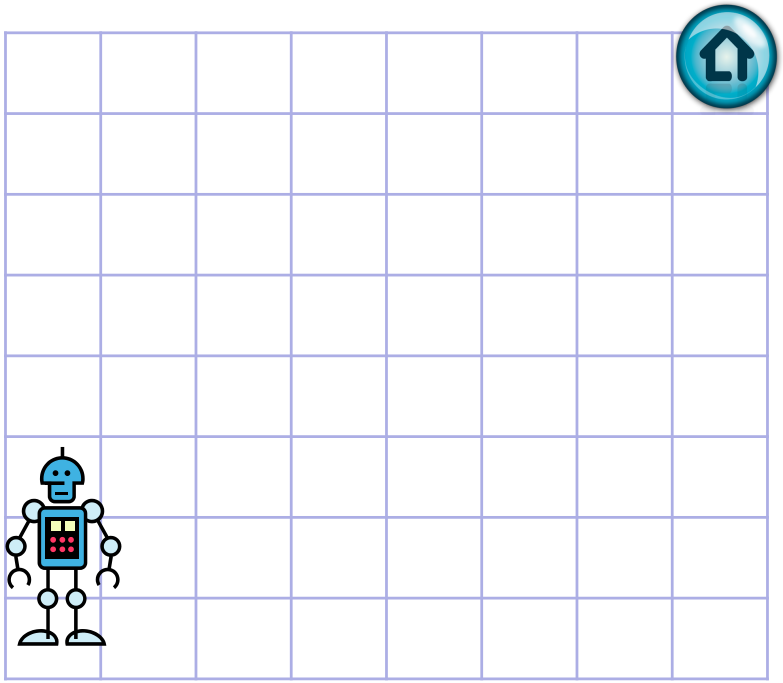
- In the worst case, no small policy
- Do online search for current action
  - May still work well, particularly when short horizon search is sufficient

Outline

- Overview
- Definitions
- Basic properties
- Intractability and Easier Subclasses
- Large state spaces
- Online search

Markov Decision Process

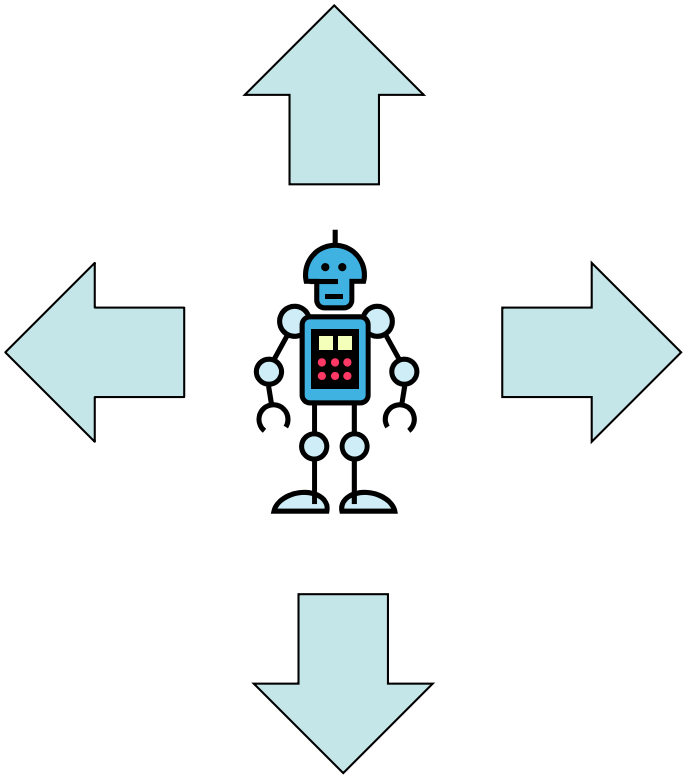
- Markov Decision Process (MDP) is defined by *<S,A,T,R>*
- State *S* : Current description of the world
  - Markov: the past is irrelevant once we know the state
  - Navigation example: Position of the robot



Actions

- MDP *<S,A,T,R>*
- Actions *A* : Set of available actions
  - Navigation example:
    - Move North
    - Move South
    - Move East
    - Move West

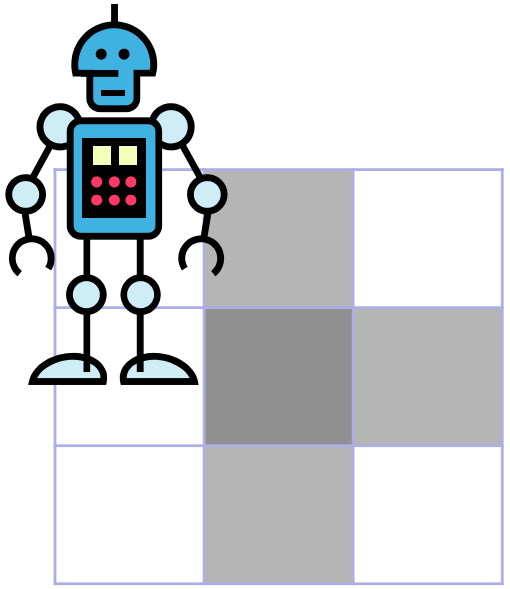
Robot navigation

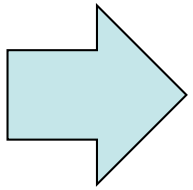


Transition Function

- MDP *<S,A,T,R>*
- Transition function *T* :
  - *T(s, a, s')* = Pr*(s' | s, a)*
  - Navigation example:
    - Darker shade, higher probability

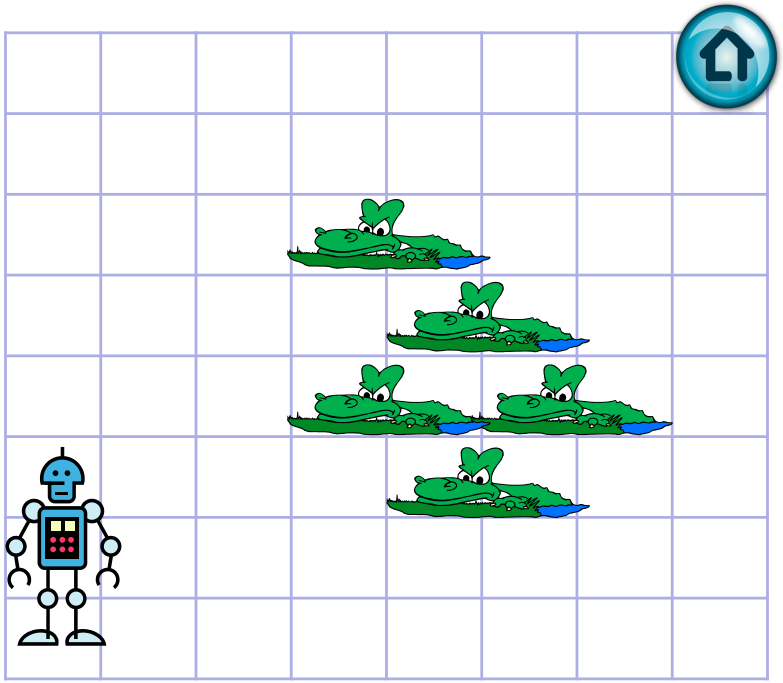
Robot navigation





Reward

- MDP *<S,A,T,R>*
- Reward function *R* : Reward received when action *a* in state *s* results in transition to state *s'*
  - *R(s, a, s')*
  - Navigation example:
    - 100 if *s'* is Home
    - -100 if *s'* is in the danger zone
    - -1 otherwise



Example of 3 state, two action Markov Decision Process *<S,A,T,R>*

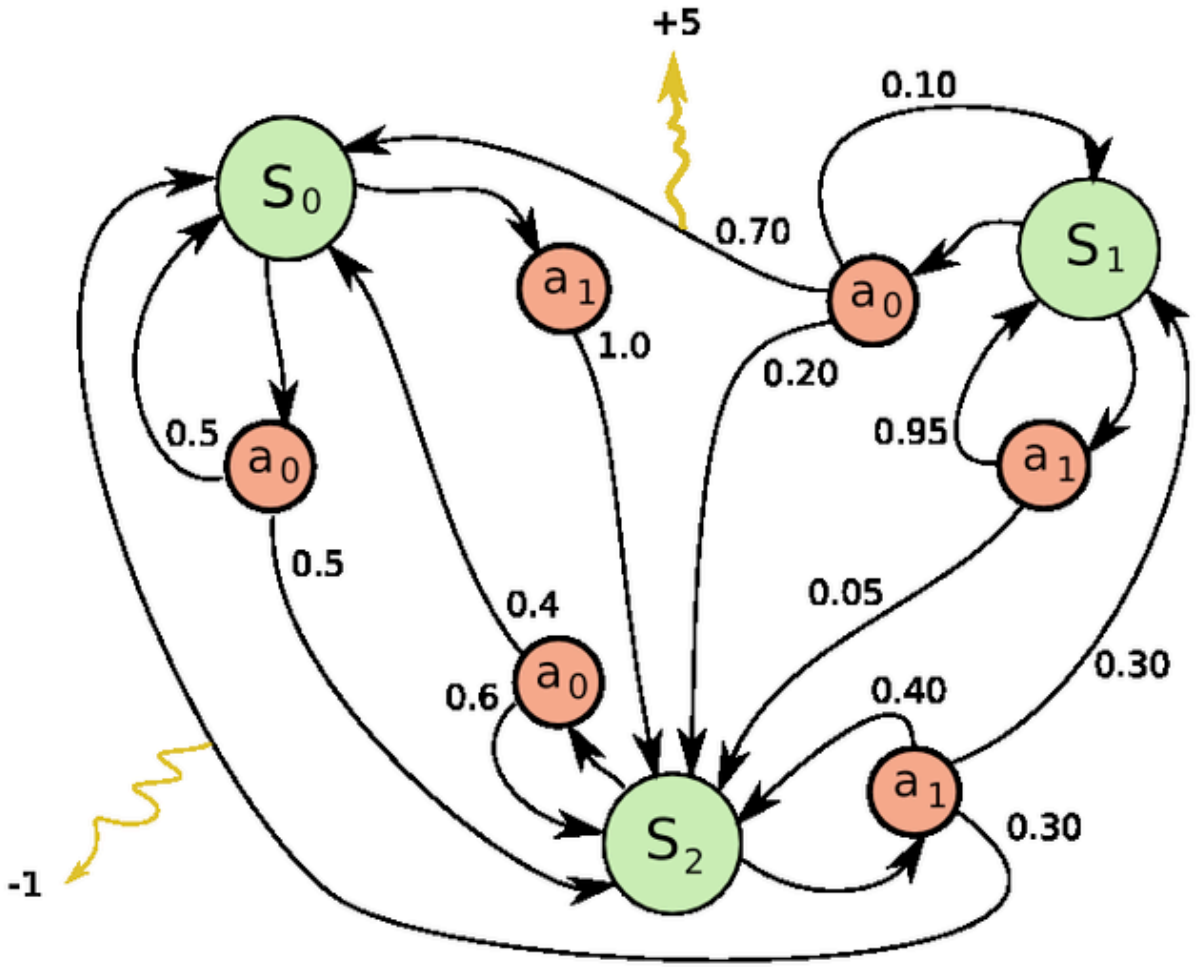
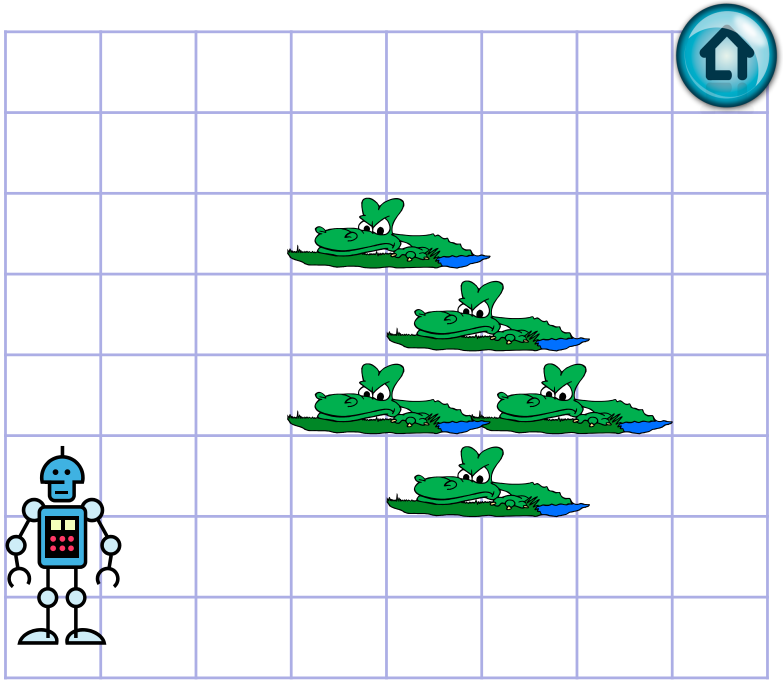


Image from http://en.wikipedia.org/wiki/File:Markov_Decision_Process_example.png

Policy

- MDP *<S,A,T,R>*
- Policy π : Function from state to action
  - $a = \pi(s)$
  - Navigation example:
    - Which direction to move at current location

Robot navigation



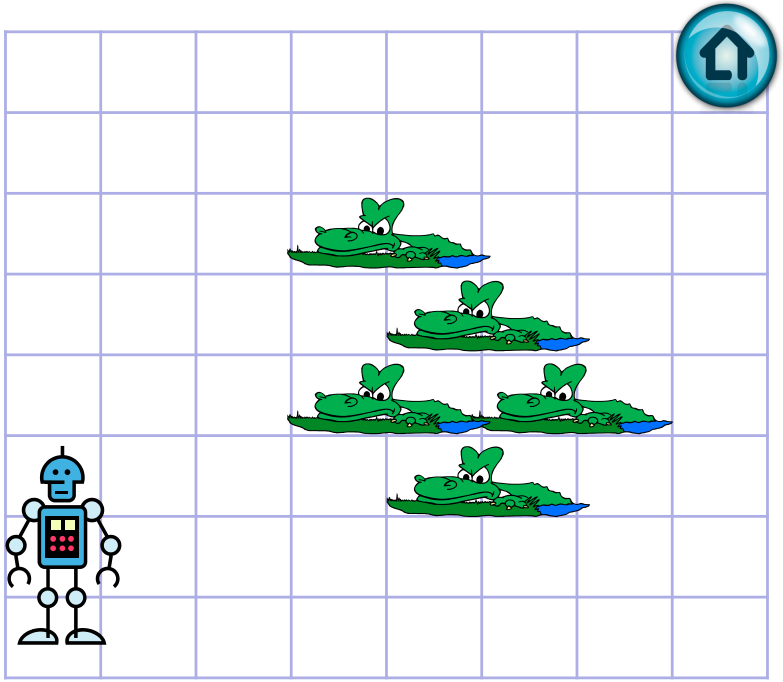
- MDP *<S,A,T,R>*
- Optimal Policy π^* : Function π that maximizes

$$\sum_{t=0}^{\infty} \gamma^t E(R(s_t, a_t, s_{t+1}))$$

$$\text{where } a_t = \pi(s_t)$$

  - Navigation example:
    - The best move at the current location

Robot navigation

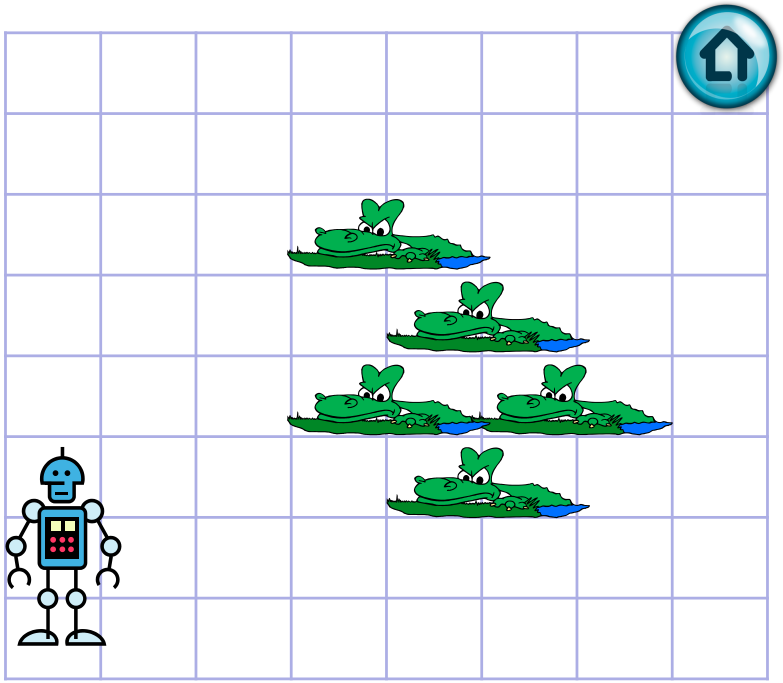


$$\sum_{t=0}^{\infty} \gamma^t E(R(s_t, \pi(s_t), s_{t+1})) \text{ where } \gamma \in (0, 1)$$

- γ is the discount factor
  - Summation finite
  - Future less important
  - Probability 1- γ process terminates

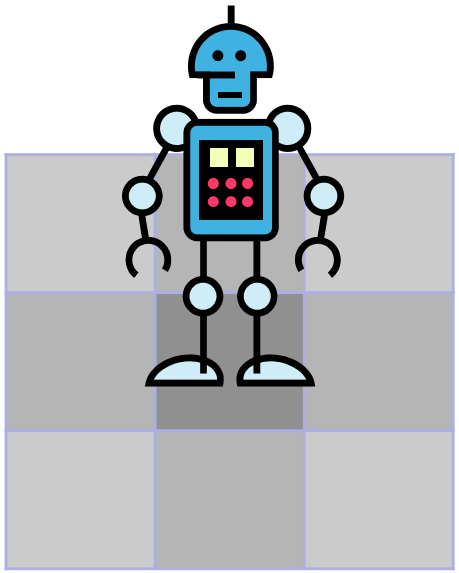
Partially Observable Markov Decision Process

- Partially Observable Markov Decision Process (POMDP) *<S,A,T,R,Ω,O>*
- Observations *Ω* : Set of possible observations
  - Navigation example:
    - Set of locations output by GPS sensor



- POMDP *<S,A,T,R,Ω,O>*
- Observation probability function *O* : Probability of observing *o* in state *s′* when previous action is *a*
  - *O(o, a, s′)* = Pr*(o | a, s′)*
  - Navigation example:
    - Darker shade, higher probability

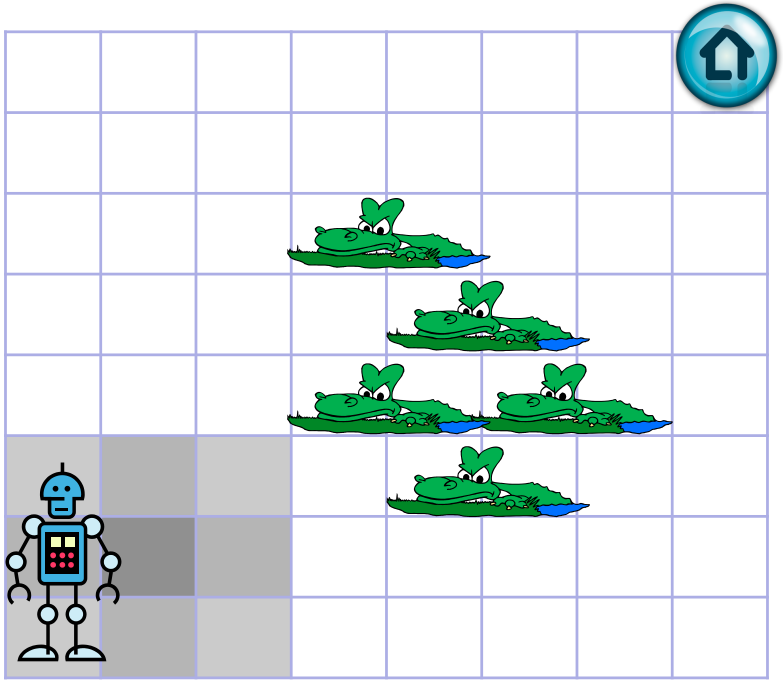
Robot navigation



Belief

- POMDP *<S,A,T,R,Ω,O>*
- Belief *b* : Probability of state *s*
  - *b(s)* = Pr(*s*)
  - Navigation example:
    - Exact position of robot unknown
    - Only have probability distribution of positions, obtained through sensor readings

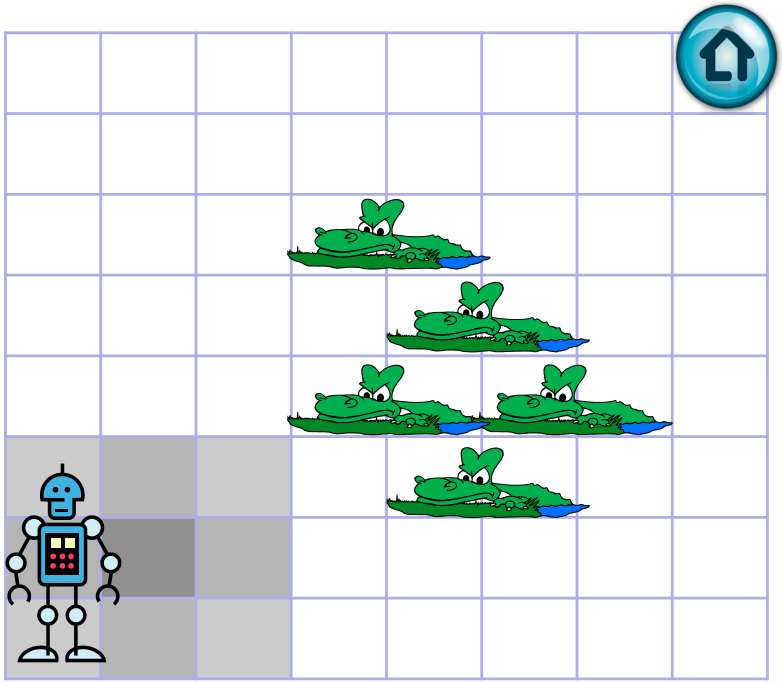
Robot navigation



POMDP Policy

- POMDP *<S,A,T,R,Ω,O>*
- Policy π : Function from belief to action
  - $a = \pi(b)$
  - Navigation example:
    - Which way to move, based on current belief

Robot navigation



- POMDP *<S,A,T,R,Ω,O>*
- *R(a,b)* : Expected reward for taking action *a* when belief is *b*

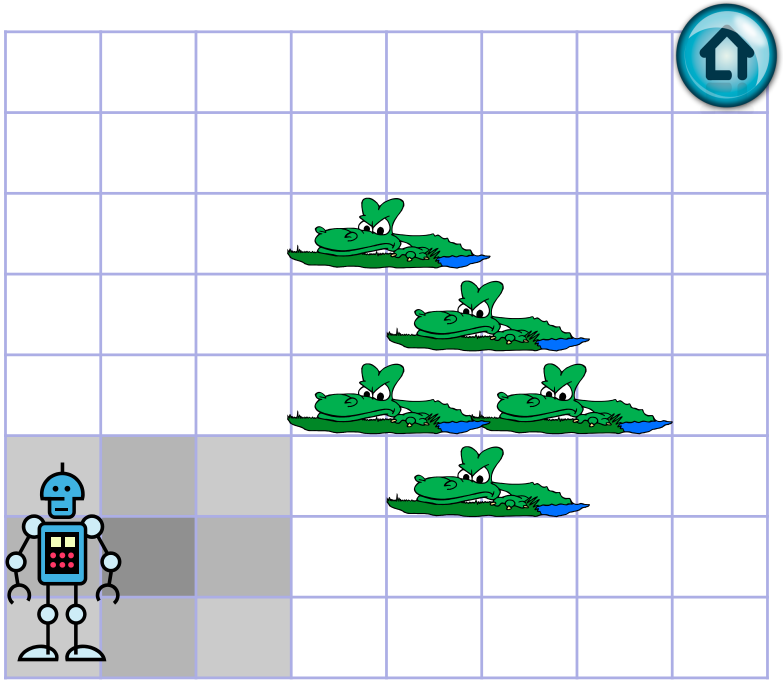
$$R(a,b) = E(R(s,a,s))$$

$$= \sum_i \sum_j T(s_i,a,s_j) b(s_i) R(s_i,a,s_j)$$

- Optimal Policy π* : Function π that maximizes

$$\sum_{t=0}^{\infty} \gamma^t R(\pi(b_t), b_t)$$

Robot navigation



Value Function

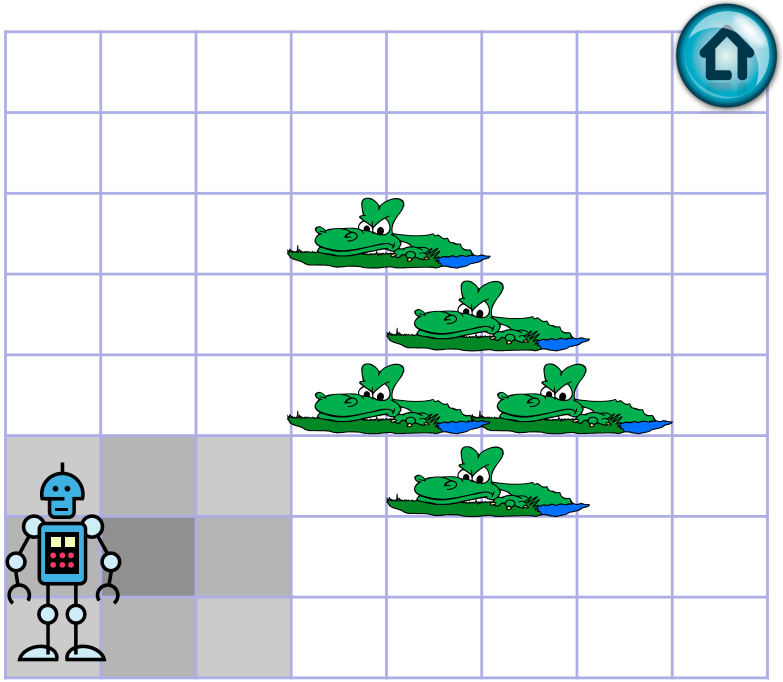
- POMDP *<S,A,T,R,Ω,O>*
- Value function for π : Expected return for starting from belief *b*

$$V^{\pi}(b) = \sum_{t=0}^{\infty} \gamma^t R(\pi(b_t), b_t),$$

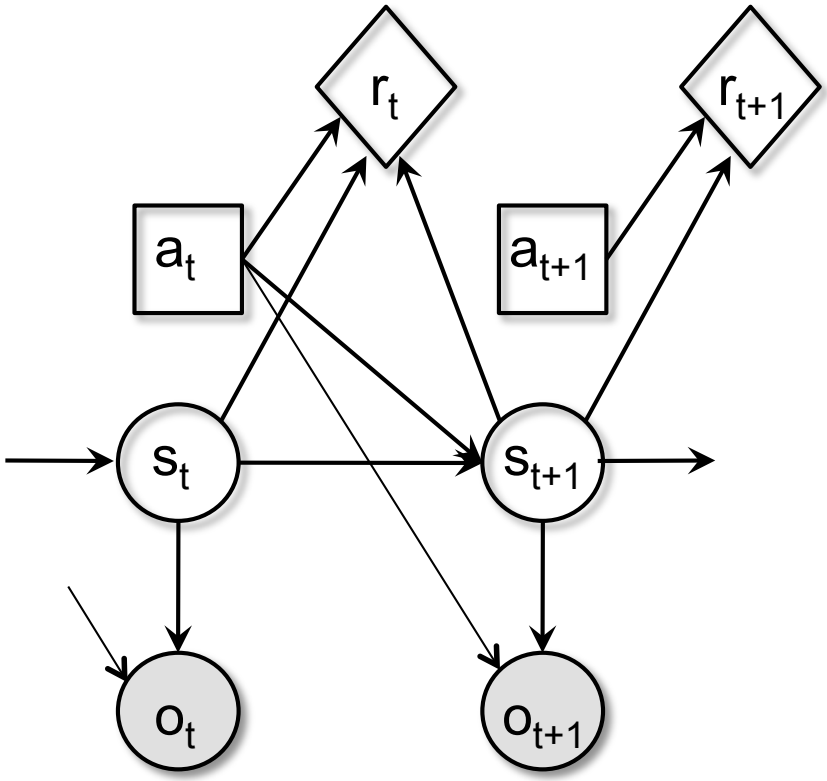
$$\text{with } b_0 = b$$

- Optimal value function *V* :* Value function associated with an optimal policy π^*

Robot navigation



POMDP as Graphical Model



Outline

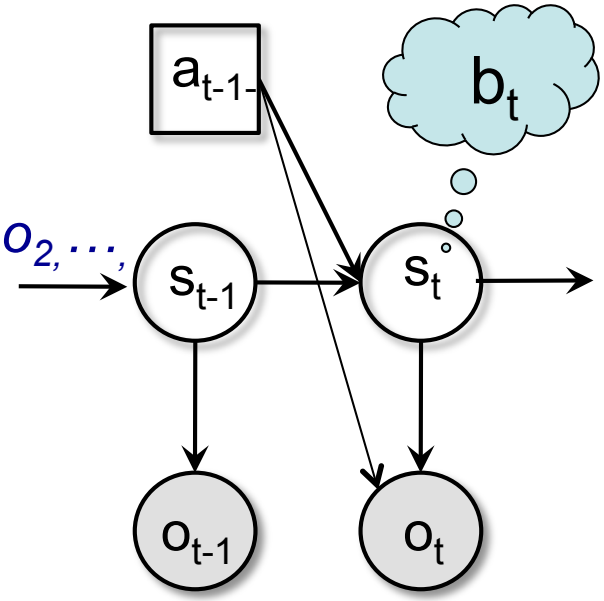
- Overview
- Definitions
- Basic properties
- Intractability and Easier Subclasses
- Large state spaces
- Online search

Outline

- Basic properties
  - b_t can be updated from b_{t-1}
  - Finite horizon value function
    - Piecewise linear convex
    - Represented as collection of α-vectors
    - Backup operator and value iteration
  - Infinite horizon value function
    - Value iteration converges
    - Convex, can be approx by α-vectors

Belief

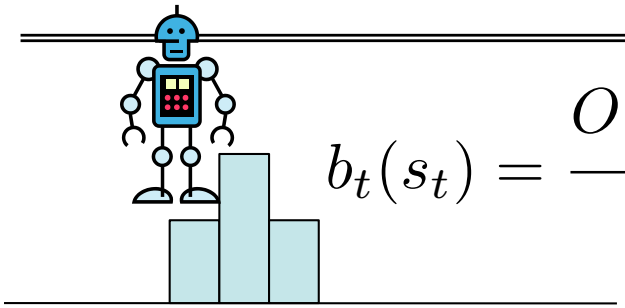
- POMDP *<S,A,T,R,Ω,O>*
- Compute belief from history of actions and observations
  $a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t$



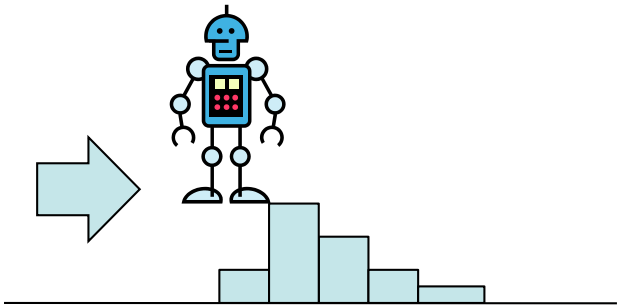
Current belief $b_t(s_t) = \Pr(s_t | a_0, \ldots, a_{t-1}, o_t)$
can be updated from previous belief using

$$b_t(s_t) = \frac{O(o_t, a_{t-1}, s_t) \sum_{s_{t-1}} T(s_{t-1}, a_{t-1}, s_t) b_{t-1}(s_{t-1})}{\Pr(o_t | a_{t-1}, b_{t-1})}$$

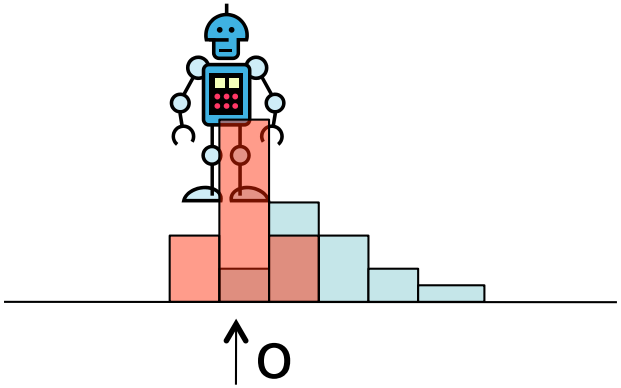
where $\Pr(o_t | a_{t-1}, b_{t-1}) = \Pr(o_t | a_0, \ldots, o_{t-1}, a_{t-1})$
is a normalizing factor to make the sum one

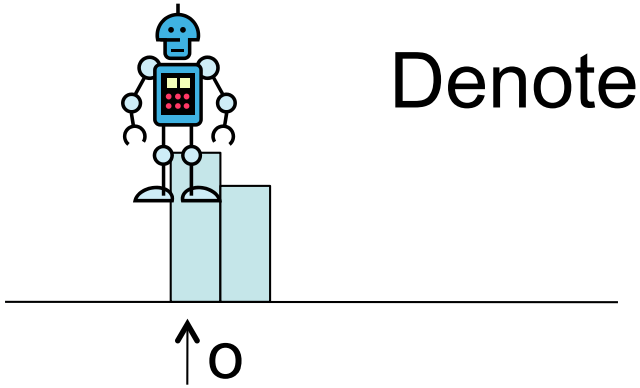


$$b_t(s_t) = \frac{O(o_t, a_{t-1}, s_t) \sum_{s_{t-1}} T(s_{t-1}, a_{t-1}, s_t) b_{t-1}(s_{t-1})}{\Pr(o_t | a_{t-1}, b_{t-1})}$$



- Markov property
- Finite sufficient statistics when state space is finite

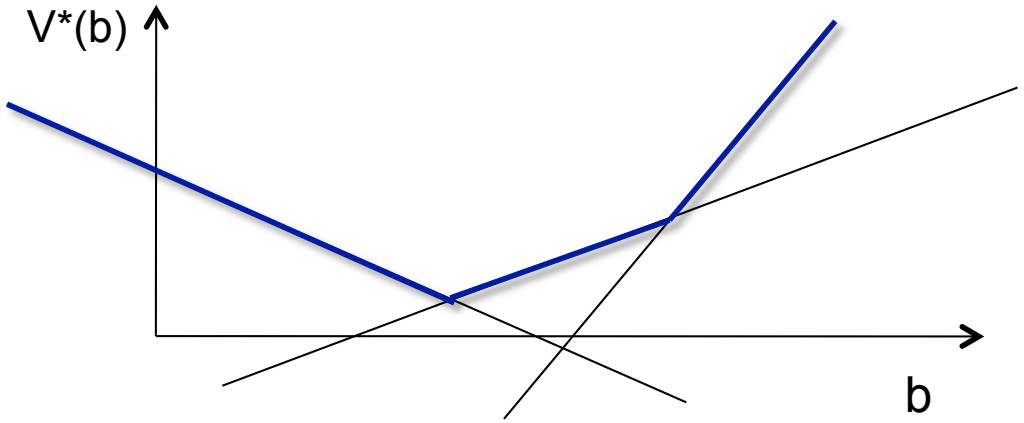




Denote $b_t = \tau(o_t, a_{t-1}, b_{t-1})$

Details

Optimal Value Function



- For finite horizon POMDP, optimal value function is piecewise linear

$$V^*(b) = \max_{\alpha \in \Gamma} \alpha \cdot b$$

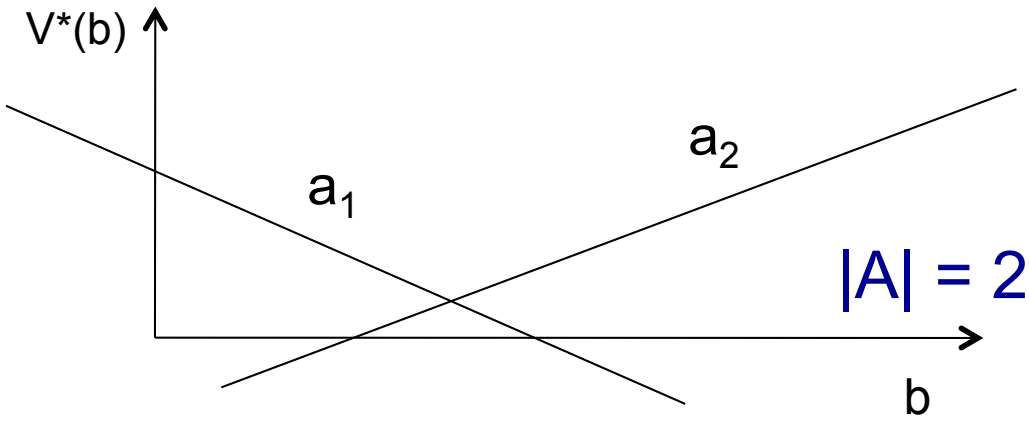
$$= \max_{\alpha \in \Gamma} \sum_i \alpha(s_i) b(s_i)$$

POMDP

Smallwood & Sondik, 1973

Finite Horizon

- The construction of $V^*(b) = \max_{\alpha \in \Gamma} \alpha \cdot b$ uses dynamic programming
- Value iteration algorithm
- Start with horizon 1 problem V_1^*
  - Can be shown to be piecewise linear with each linear function corresponding to an action

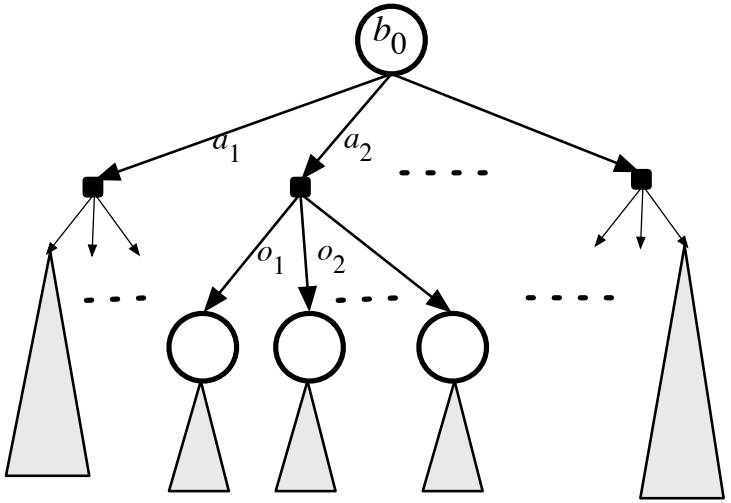


|A| = 2, gives 2 α-vectors

- Construct V_i^* out of V_{i-1}^* by applying backup operator H to get $V_i^* = HV_{i-1}^*$

$$V_i(b) = HV_{i-1}(b)$$

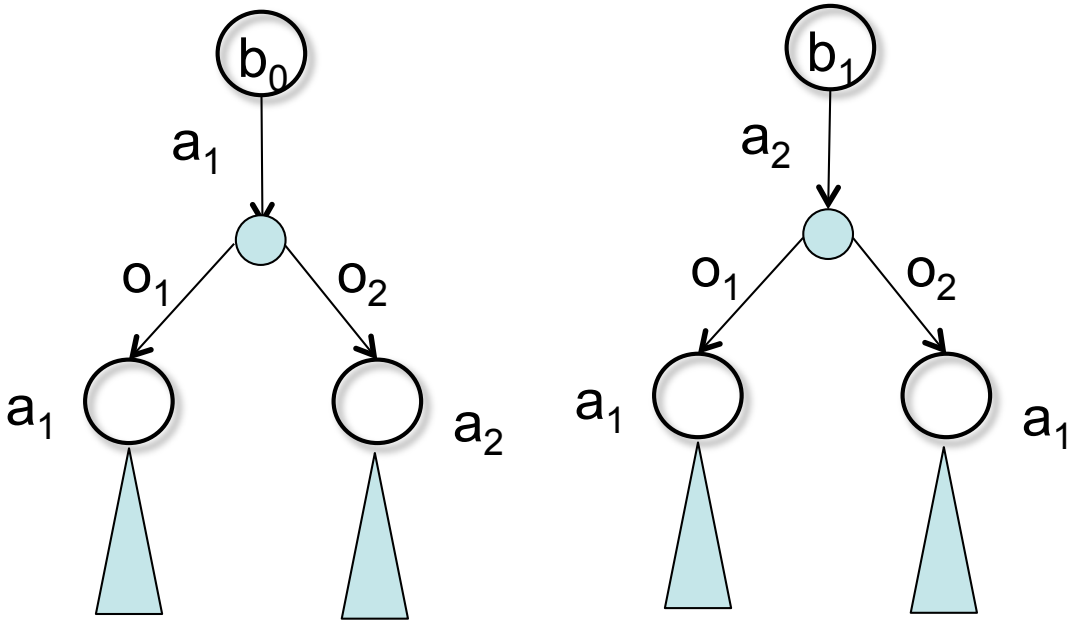
$$= \max_a R(a,b) + \gamma \sum_o p(o|a,b) V_{i-1}(\tau(o,a,b))$$



Can show that $V_i^*(b)$ max of linear functions if $V_{i-1}^*(b)$ max of linear functions

Details

- Unfortunately, for horizon *k* problem, the number of functions in Γ may grow double exponentially, approx $|A|^{|\Omega|^{k-1}}$



…

- Each α-vector correspond to a policy tree.
- Different tree may work best for different belief

Infinite Horizon

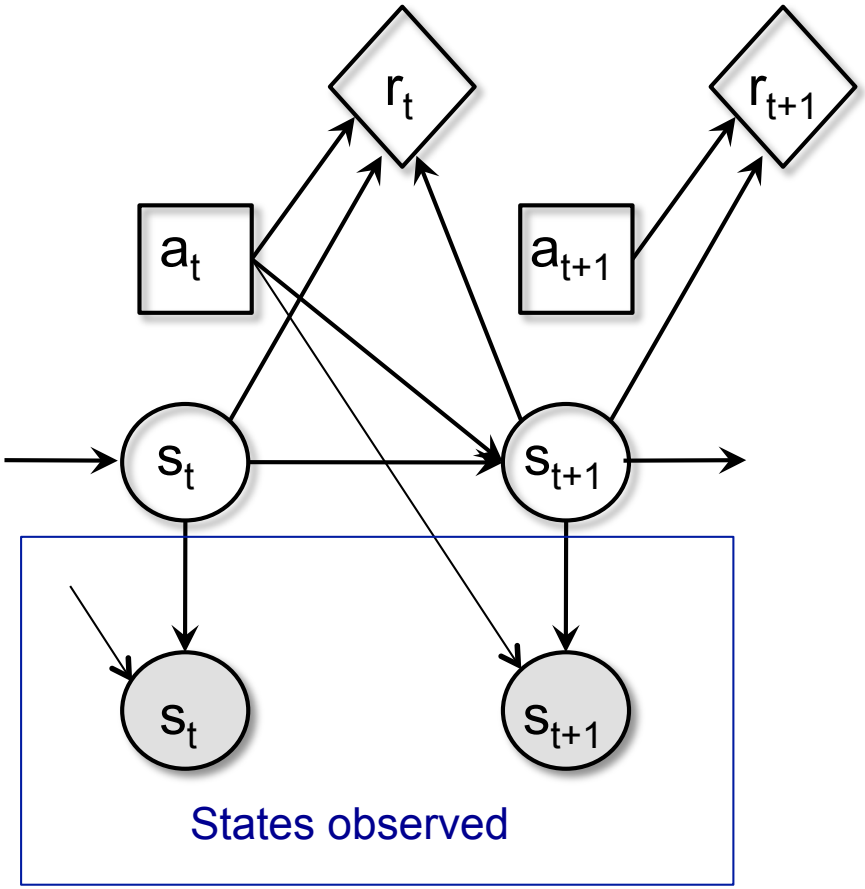
- For finite horizon POMDP, optimal value function is piecewise linear
- Taking the horizon *k* to infinity,
  - Value iteration converges to unique convex function, regardless of initialization
  - Value function can be approximated arbitrarily closely by finite number of α-vectors
  - Value function satisfies the Bellman optimality equation

Details

$$V^*(b) = \max_a R(a,b) + \gamma \sum_o p(o|a,b) V^*(\tau(o,a,b))$$

Value Function for MDP

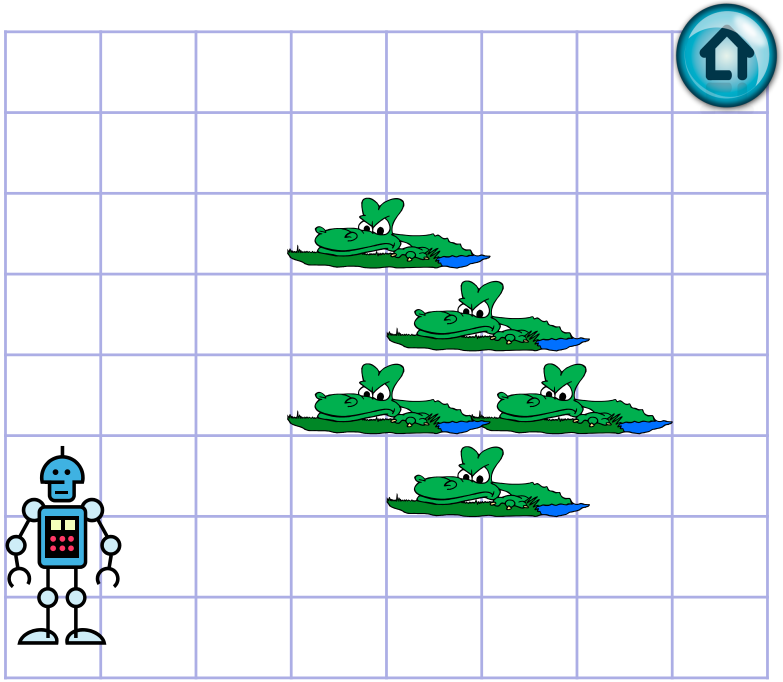
- MDP can be considered a special case of POMDP where the state is observed
- Belief is zero everywhere except at one state



NUS National University of Singapore

- *V*(s)* is a function of the state
  - Value iteration practically effective when state space is not too large
  - *V*(s)* can be found in time polynomial in */S/* using linear programming

Robot navigation



Value function is a vector of length 64

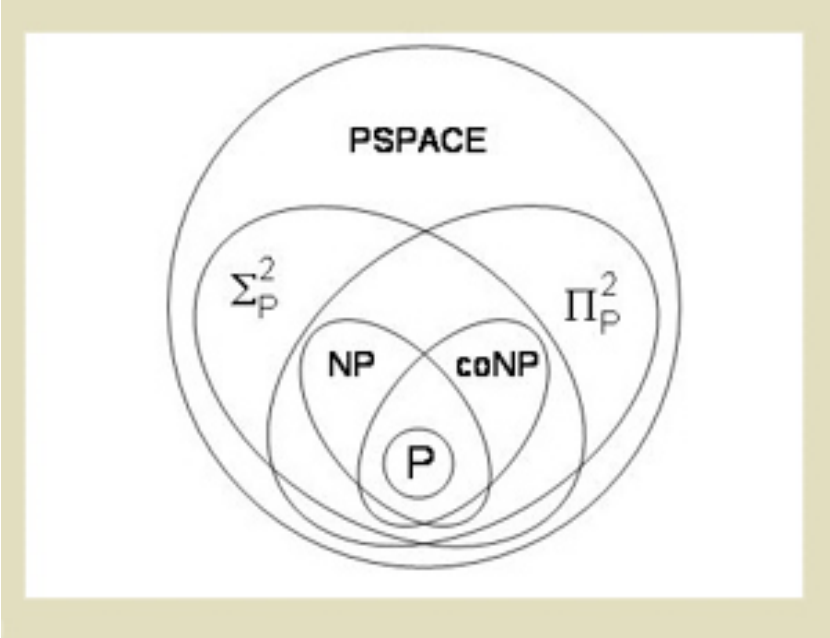
Outline

- Overview
- Definitions
- Basic properties
- Intractability and Easier Subclasses
- Large state spaces
- Online search

Outline

- Intractability and Easier Subclasses
  - Worst case: PSPACE complete and no small policy
  - Easier cases: achievable using point-based methods
    - Small reachable belief space: poly time approximation
    - Small optimal reachable space: small policy
  - Useful properties indicating smaller belief space

Intractability

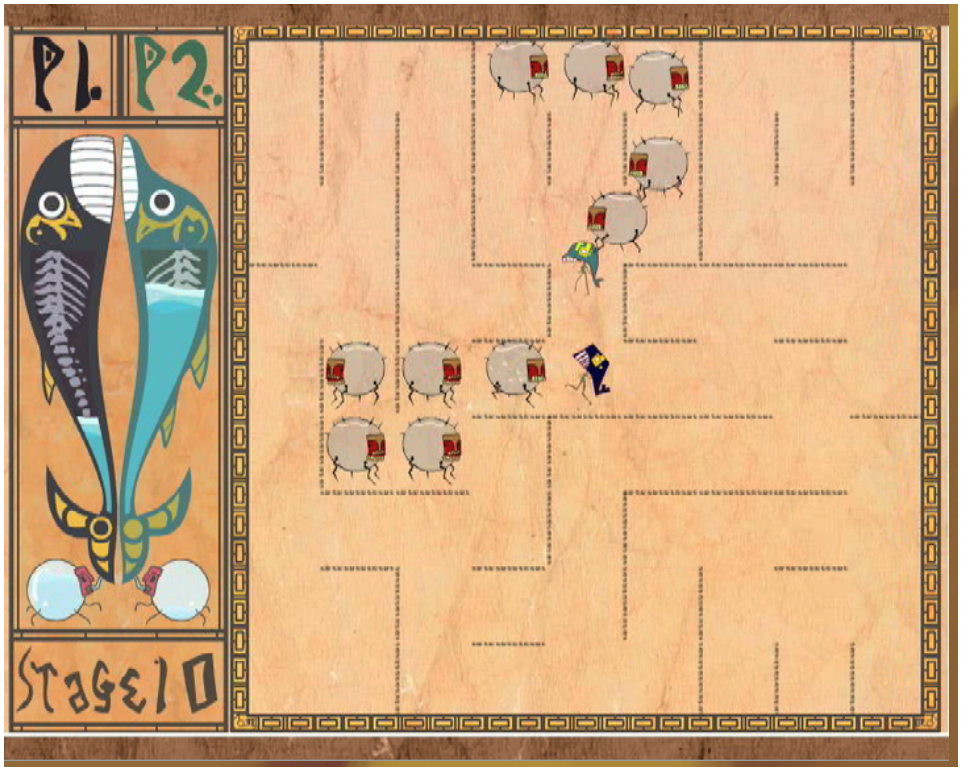


Papadimitriou and Tsitsiklis, 1987

- Finite horizon POMDP is PSPACE-complete
  - Computing the optimal solution intractable in the worst case
- No polynomial-sized policy that can compute optimum decision in polynomial time, assuming $PSPACE \neq \sum_2^P$
  - In the worst case, cannot hope to have a small optimal policy that can be executed quickly, even if willing to spend a long time to find the policy

Image from http://ocw.mit.edu/courses/mathematics/18-405j-advanced-complexity-theory-fall-2001/

Maybe your problem is not so hard …



- Aim: Human player and AI must lure the monsters and smash them …
- Idea: Figure out human intention, then cooperate
  - Human intention as unobserved variable, inferred through observing human action
  - POMDP

But, ….. turns out that assuming human is optimal works well

- Deterministic problem
- No need to model uncertainty

POMDP

Ngo, 2011

QMDP

- MDP can be efficiently solved when state space is not too large
- Can efficiently compute

$$Q_{MDP}(s,a) = E(R(s,a,s')) + \gamma \sum_{s'} T(s,a,s')V(s')$$

$$Q(a,b) = \sum_{s} b(s) Q_{MDP}(s,a)$$

- QMDP selects action that maximizes Q(a,b)
  - Assume state uncertainty is gone after one step

Littman, Cassandra, Kaelbling, 1995

NUS National University of Singapore

- Example application: Aircraft Collision Avoidance

	QMDP	TCAS
Pr(NMAC)	6.98×10^{-5}	1.43×10^{-4}
Pr(Alert)	2.01×10^{-4}	5.03×10^{-4}

Kochenderfer & Chryssanthacopoulos, 2011

Video from http://www.youtube.com/watch?v=-tIcWObSk8I&feature=player_embedded

- QMDP-type methods can fail when need to take account of future uncertainty: Coastal navigation

POMDP

Videos from http://robots.stanford.edu/videos.html

Roy, Gordon, Thrun, 2004

NUS National University of Singapore

Examples: QMDP fails compared to belief space policies

- Tag
  - Robot find target that moves away from it
  - Robot knows own position but not position of target until at the same location
  - 870 states
  - Value: QMDP -16.6, PBVI -6.75

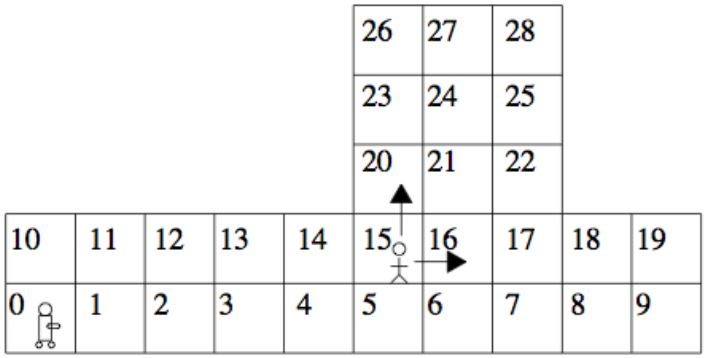


Image and results from Pineau, Gordon, Thrun, 2006

- RockSample
  - Robot needs to collect samples of good rock
  - Rock value unknown, can be sensed using noisy sensor
  - Robot position known
  - 12,545 states
  - Value: QMDP 0, HSVI2 20.6

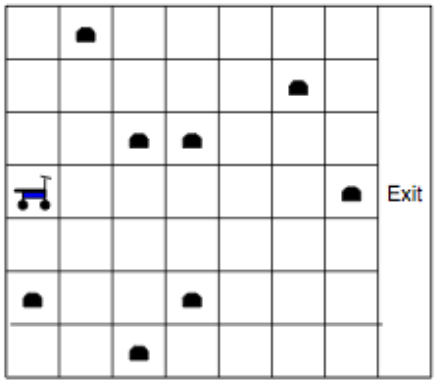


Image and results from Smith & Simmons, 2005

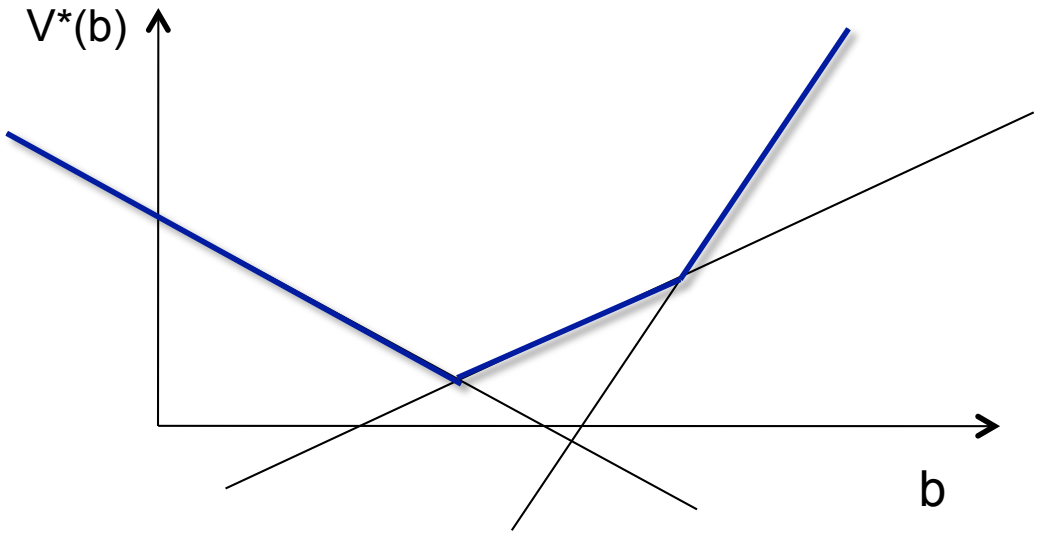
POMDP

Easier to Approximate Subclasses

- POMDP intractable in the worst case
- Aim: do well when the problem is actually not so hard ….
- Hope: many problems of practical interest are actually not so hard ….
  - if only we knew how to represent them and search effectively for solutions

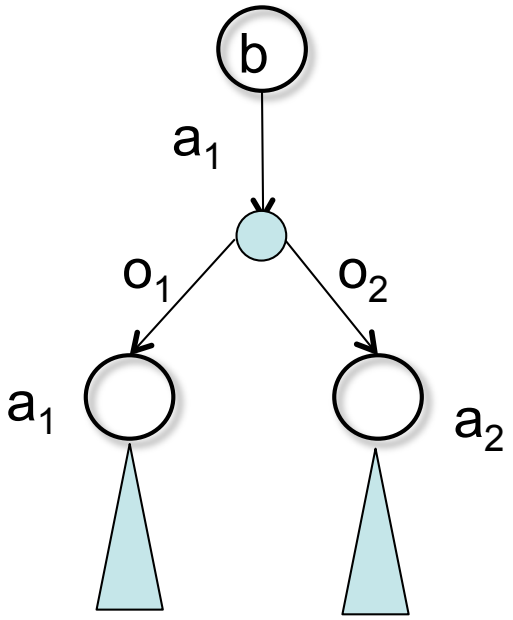
Point Based Methods

- Point-based methods give some of current state-of-the-art solvers
- Use α-vectors to represent a piecewise linear convex value function
- Use a variant of value iteration where backup is done only at selected beliefs

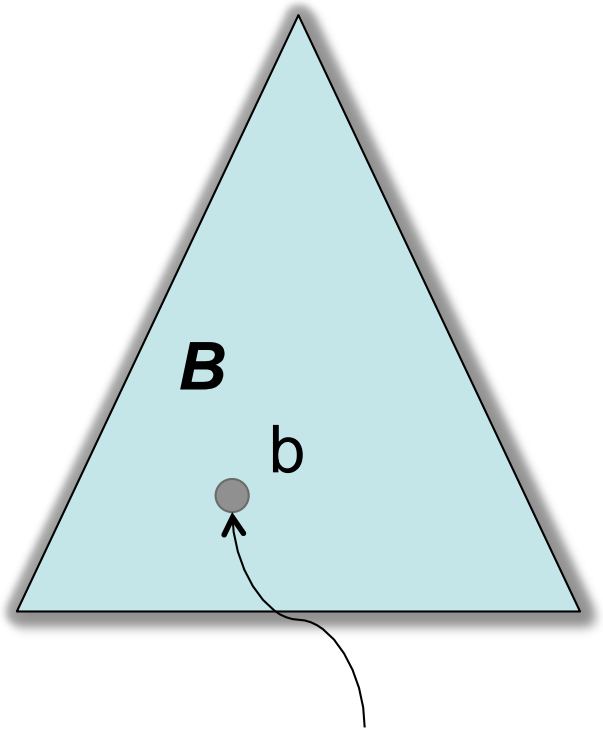


PBVI (Pineau, Gordon, Thrun, 2006), Perseus (Spann, Vlassis, 2005), HSVI (Smith & Simmons, 2004, 2005), SARSOP (Kurniawati, Hsu, Lee, 2008), etc.

- At point *b*, point-based backup gives
  - Best α-vector at *b* that can be constructed from current Γ
  - Best policy tree at *b* that can be by extending current policy trees



- Each point-based backup at b adds one α-vector
  - Compare to double exponential growth in α-vectors for exact backup at all beliefs in the domain
  - Resulting α-vector optimal only in the neighbourhood of b
  - Computational cost $O(|A||\Omega||\Gamma_{i-1}|)$



Backup at single point rather than whole space

NUS National University of Singapore

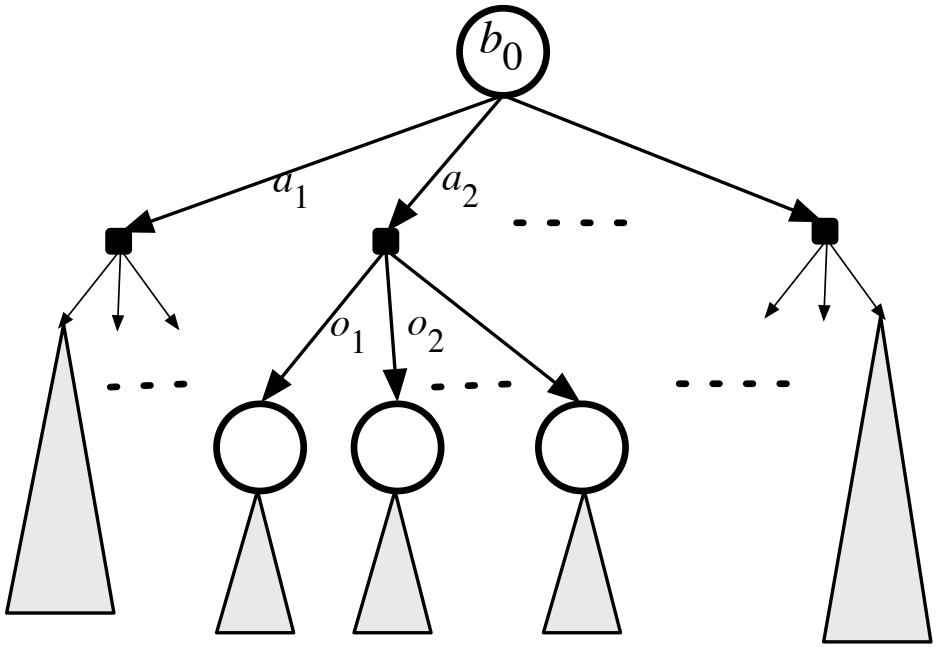
- During point-based backup
  - Store action responsible for construction of the α-vector
- When execution of the policy
  - At *b*, run action associated with α-vector responsible for

$$V(b) = \max_{\alpha \in \Gamma} \alpha \cdot b$$

- Important: how to select points for doing point-based backup?

Reachable Beliefs

- For problems like Tag and RockSample, we know the initial belief b_0
- Let $B(b_0)$ be the space reachable from b_0 under arbitrary actions and observations
- Only care about $B(b_0)$, so do point-based backup only at a finite subset of $B(b_0)$

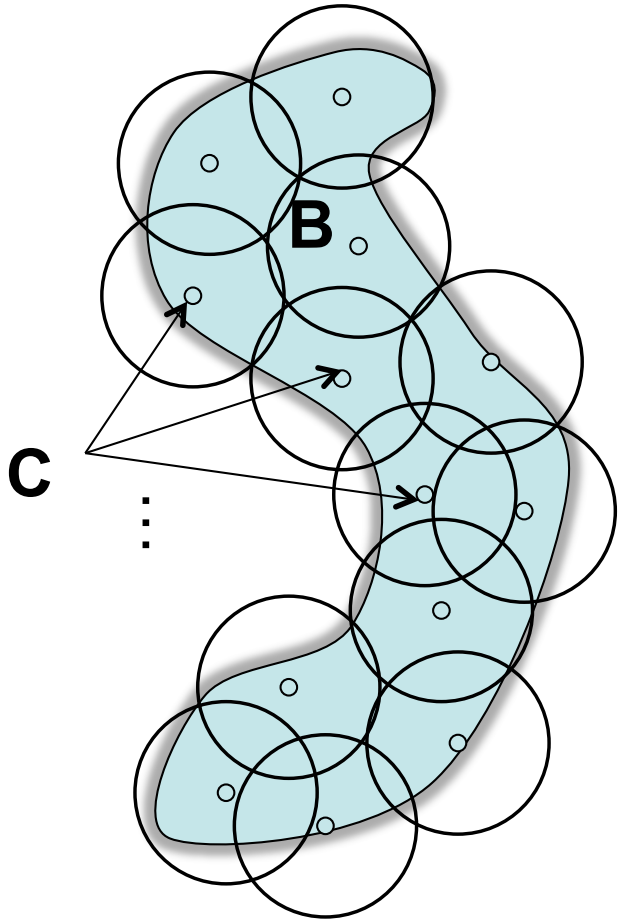


- If the selected points for backup "covers" $B(b_0)$ well, approximation error will be small
- If the number of selected points need to "cover" $B(b_0)$ is small, computationally efficient.

Covering Number

- Measure the "size" of belief space
- A *δ-cover* of a set $B \subseteq X$ is a set of point $C \subseteq X$ such that for every point $b \in B$, there is a point $c \in C$ with $||b-c|| < \delta$
- The *δ-covering number* of *B*, denoted by *C(δ)*, is the size of the smallest *δ*-cover of *B*
- Use l_1-metric on beliefs

$$||b_1 - b_2||_1 = \sum_i |b_1(s_i) - b_2(s_i)|$$



Small Reachable Belief Space

- POMDP can be efficiently approximated when reachable belief space $B(b_0)$ is small
- Specifically, policy with $|V^*(b_0) - V(b_0)| \leq \epsilon$ can be found using a point based method in time

$$O\left(\mathcal{C}\left(\frac{(1-\gamma)^2\epsilon}{8\gamma R_{\max}}\right)^2 \log_\gamma \frac{(1-\gamma)\epsilon}{2R_{\max}}\right)$$

Hsu, Lee, Nan, 2007

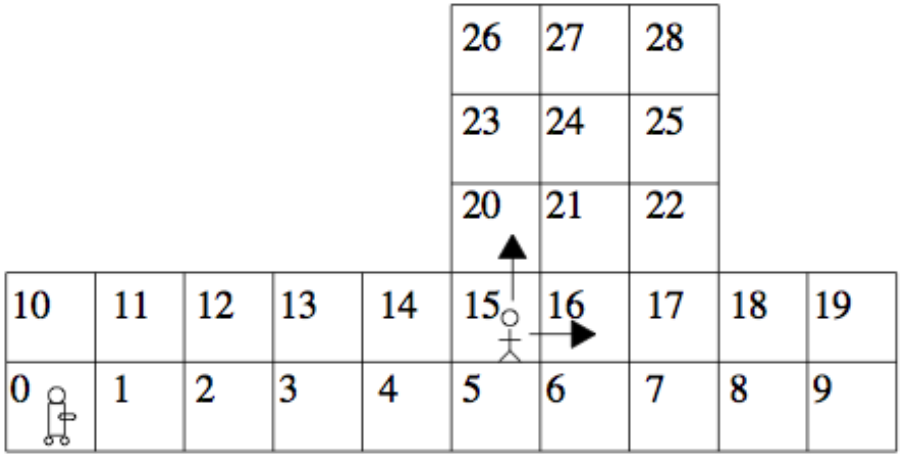
Details

Properties Affecting Covering Number

- Simple discretization gives an upper bound for covering number of $(|S|/\delta)^{|S|}$
  - Exponential growth with state space size
- However, problem may be significantly easier if it has
  - Subset of variables fully observed
  - Sparse beliefs
  - Factored beliefs
  - Smooth beliefs
  - Structure in transition matrices

Mixed Observability Markov Decision Process (MOMDP)

- In many practical applications, some state variables can be observed accurately
  - In Tag, belief can be treated as pair (s_r, b_p) where
    - $s_r \in \{0,\ldots,28\}$ is observed robot position *and*
    - b_p is a 29-dimensional belief of person position

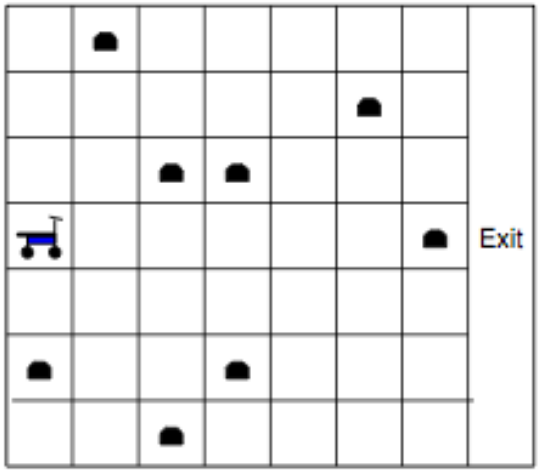


POMDP

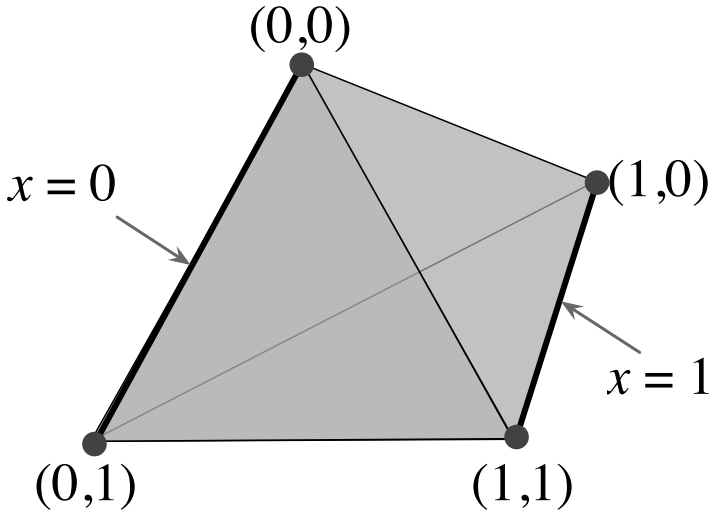
Ong, Png, Hsu, Lee 2010

– In RockSample, belief can be treated as pair (s_r, b_r) where

- $s_r \in \{0,\ldots,49\}$ is observed robot position *and*
- b_p is a 256-dimensional belief of rock properties



- Let
  - S_o be the states that the observed state variables can take,
  - S_u be the states that the unobserved state variables can take
- Belief space is union of $|S_o|$ belief spaces, each of dimension $|S_u|$
  - Covering number upper bounded by



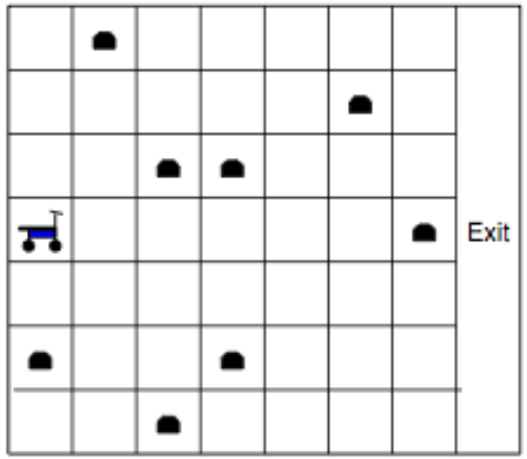
$$|S_0|(|S_u|/\delta)^{|S_u|} \text{ instead of } (|S_o||S_u|/\delta)^{|S_o||S_u|}$$

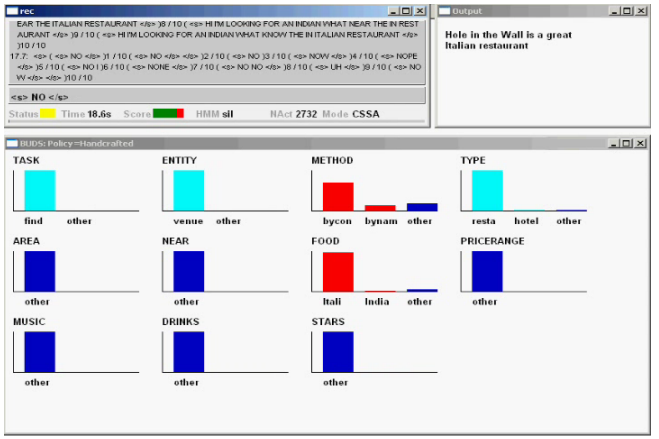
- Summary: POMDP with some state variables fully observed (MOMDP) may have exponentially smaller covering number
- For MOMDP, α-vectors of length $|S_u|$ also sufficient, further saving computation

Sparse beliefs, where beliefs can always be well approximated by a small number of non-zero coefficients, can have similar effect on the covering number

Factored Beliefs

- RockSample has 8 rocks, each which can be good or bad
  - Rock properties can take 2^8=256 states, but
  - Rock properties belief b_p can be factored and represented using 8 numbers, $p_1,\ldots,p_8$ where p_i represents the probability that rock i is good
  - Belief remain factored even after sensing
- Slot filling dialog systems often have factored belief





POMDP

- Covering number of belief space can be bounded in terms of covering number of parameter space for factored belief
  - Roughly 8 dimensional space, instead of 256 dimensional space for RockSample

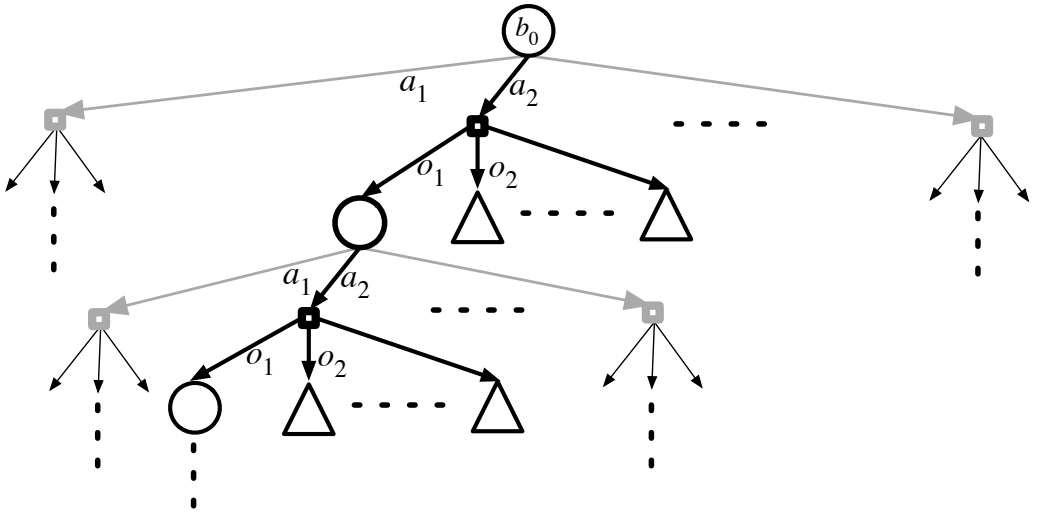
Similar for other smooth belief spaces that can be well approximated using few parameters

Ko, Ong, Hsu, Lee 2010

Space Reachable under an Optimal Policy

- Very few practical applications have small reachable belief space
- In practice,
  - use heuristics to search useful parts of belief space first, and
  - branch and bound techniques to eliminate unnecessary search
- But these will not be effective if there is no small policy
  - Recall that there is no small policy in the worst case
- Give sufficient condition for small policy?

- Let π^* be an optimal policy and $B_{\pi^*}(b_0)$ be the space reachable under π^*

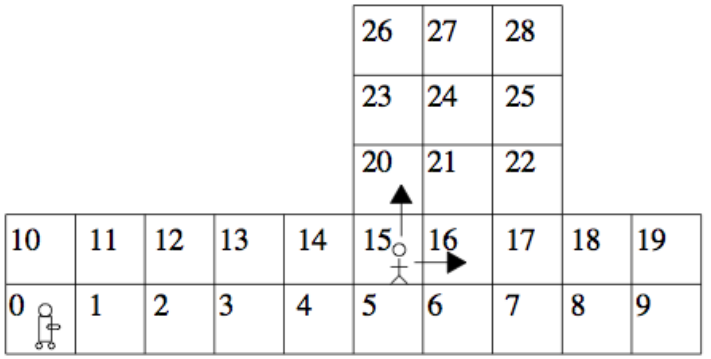


- Let π^* be an optimal policy and $B_{\pi^*}(b_0)$ be the space reachable under π^*. Let $C(\delta)$ be the covering number for $B_{\pi^*}(b_0)$. The there exists policy with error less than ε and of size

$$O\left(\mathcal{C}\left(\frac{(1-\gamma)^2\epsilon}{4\gamma R_{\max}}\right)\log_\gamma \frac{(1-\gamma)\epsilon}{2R_{\max}}\right)$$

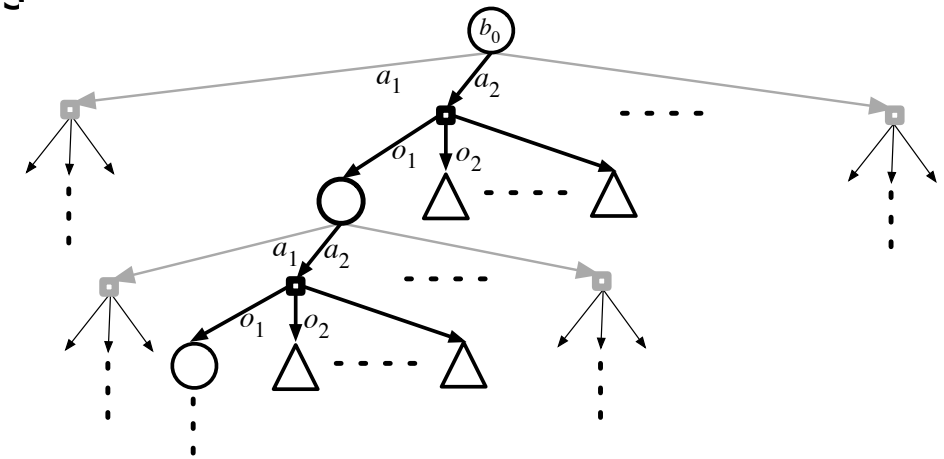
Hsu, Lee, Nan, 2007

- Are small policies likely to exist for practical problems?
  - Tag: Policy is essentially a path through the room, terminated when person is found
- Policy small
  - when do not need to branch a lot on observations, or
  - always branch to similar beliefs



What if no good small policy?

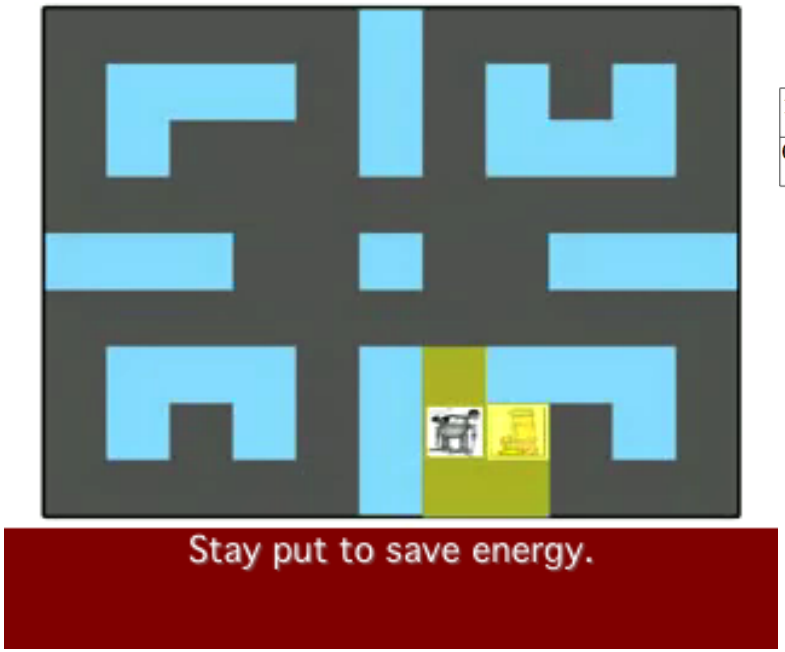
- Online search (later)



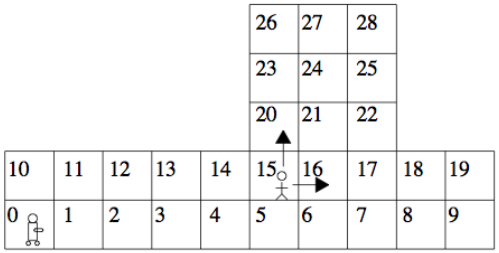
- Recap: State-of-the-art point based POMDP solvers
  - Sample points starting from b_0
    - Only consider reachable belief space from b_0
  - Incrementally adds points using latest information
    - Uses heuristic to guide search, and branch and bound to eliminate unnecessary search and get to an optimal reachable space
  - Exploits small reachable space and small optimal reachable space

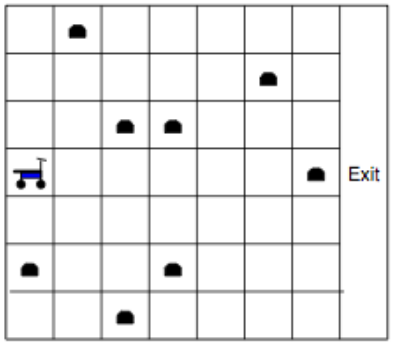
SARSOP

- Software available at http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/

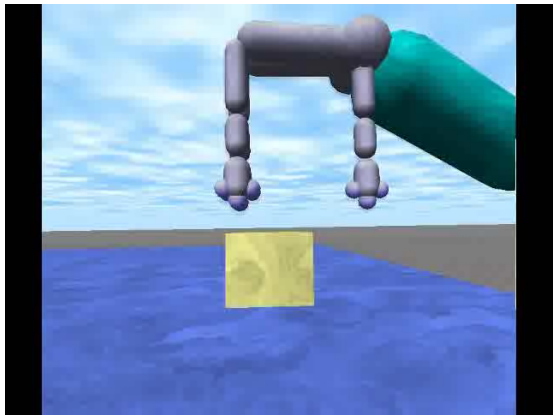


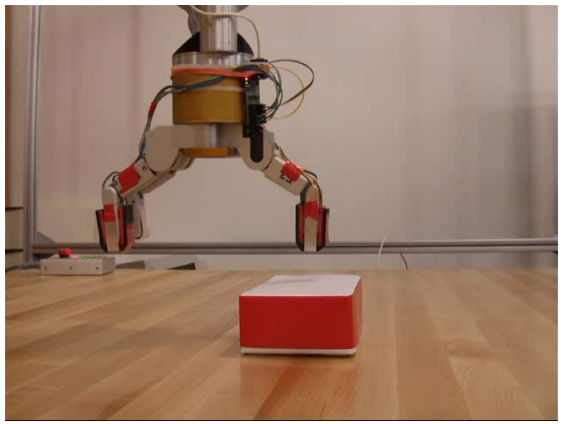
Kurniawati, Hsu, Lee, 2010





POMDP





Grasping videos from http://people.csail.mit.edu/kjhsiao/abstractstatepomdps/index.html

Outline

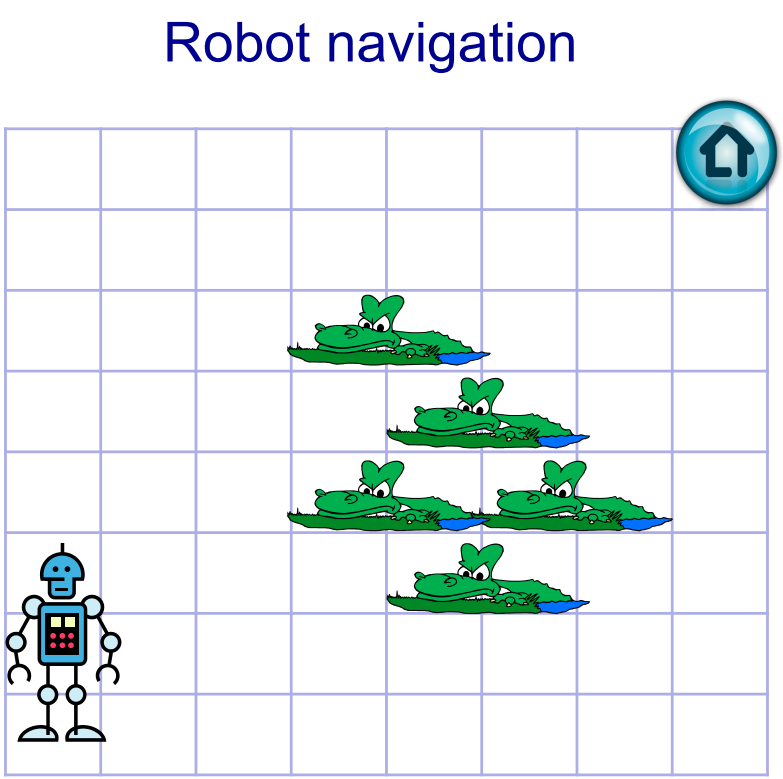
- Overview
- Definitions
- Basic properties
- Intractability and Easier Subclasses
- Large state spaces
- Online search

Outline

- Large state spaces
  - Factored POMDP
  - ADD-based method
  - Monte Carlo Value Iteration

Large State Spaces

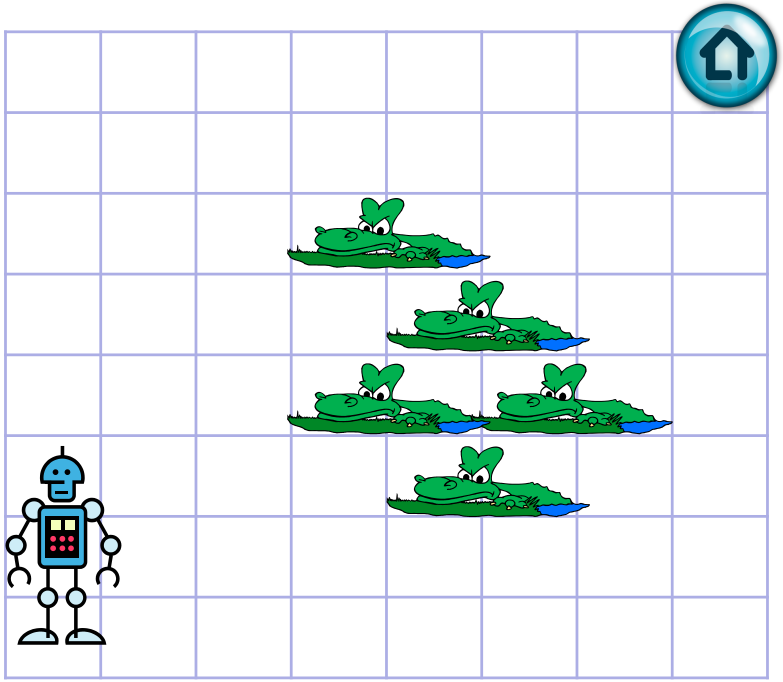
- Assume that the crocodiles in the robot navigation problem are able to move.
- State can be described by tuple $(r, c_1, \ldots, c_m)$ where r is robot position and c_i is position of crocodile i.
  - State space size 64^{m+1}
  - Grows exponentially with number of variables



Large State Spaces

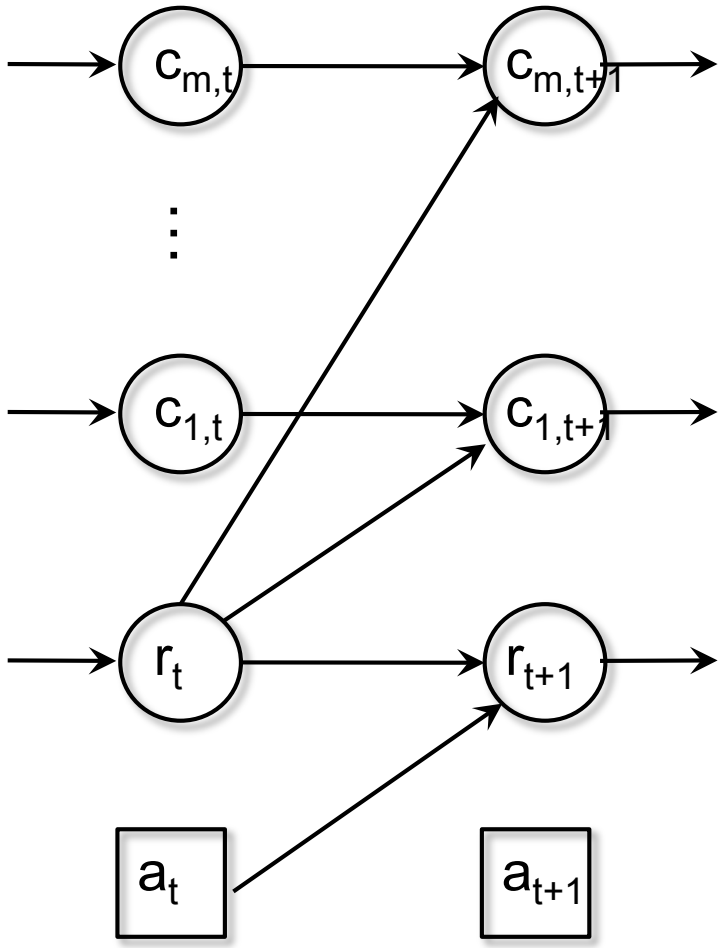
- Issue:
  - Problems are naturally described in terms of variables
  - But state space size grows exponentially in the number of variables.

Robot navigation



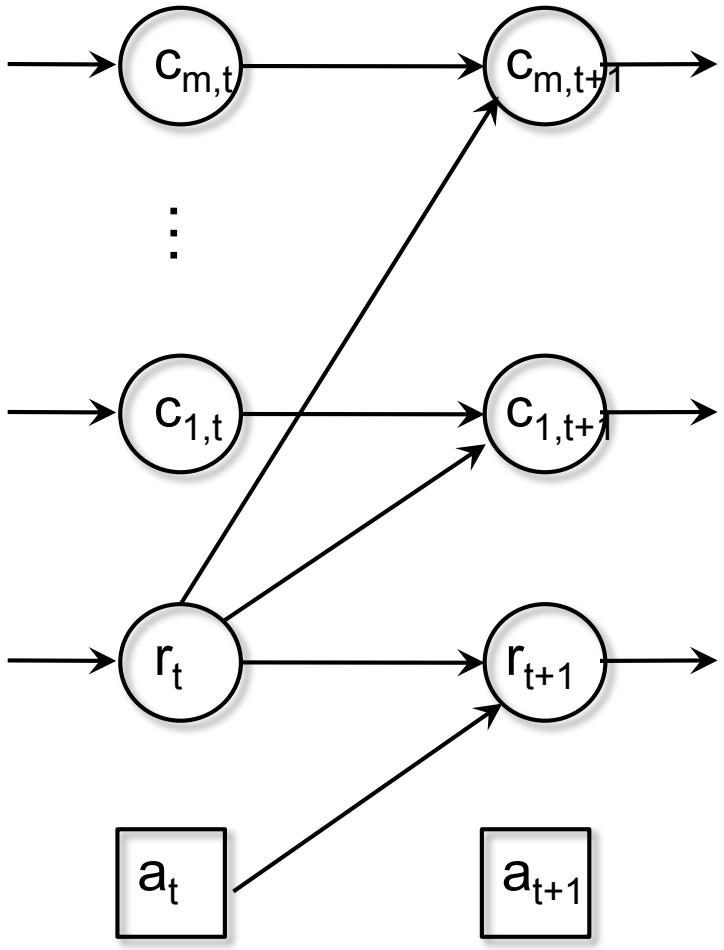
Problem Representation

- Exploit limited dependencies in transition function to get small representation for *T(s,a,s')*
  - Crocodile movement depends only on robot position and current crocodile position
  - Robot next position depends only on current position and action

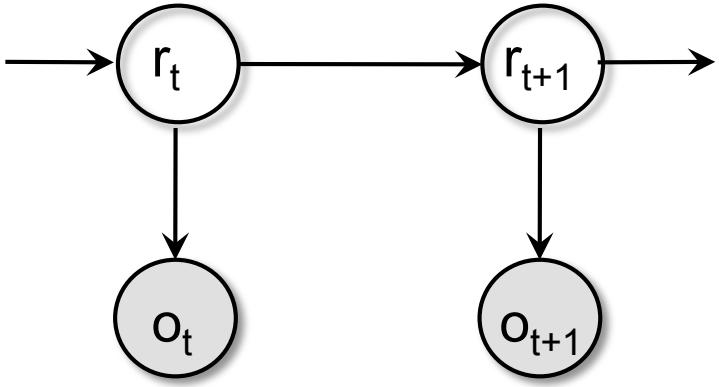


POMDP

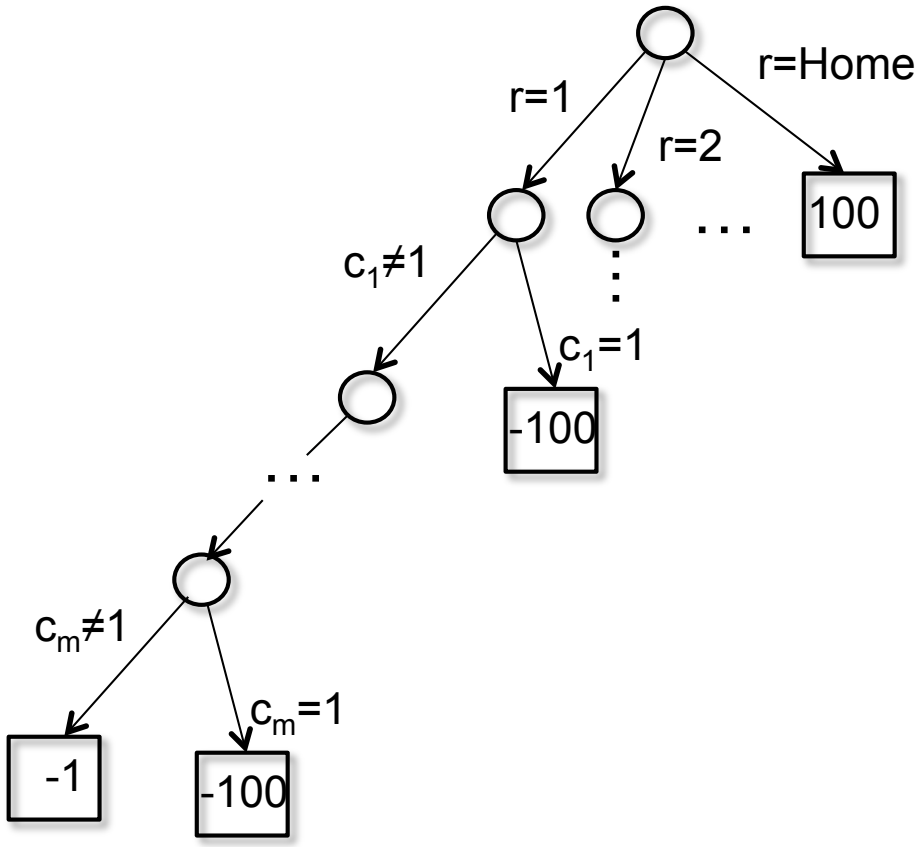
- *T(s,a,s')* can be specified by
  - $Pr(c_i'|r,c_i)$ for each *I*, and
  - *Pr(r'|a,r)* for each *a*
- Exponentially smaller probability tables



- Similarly, observation (GPS reading) depends only on position of robot, *Pr(o|r)*, instead of the whole state
- Dynamic Bayesian Network (DBN) representation gives compact represention of transition and observation functions

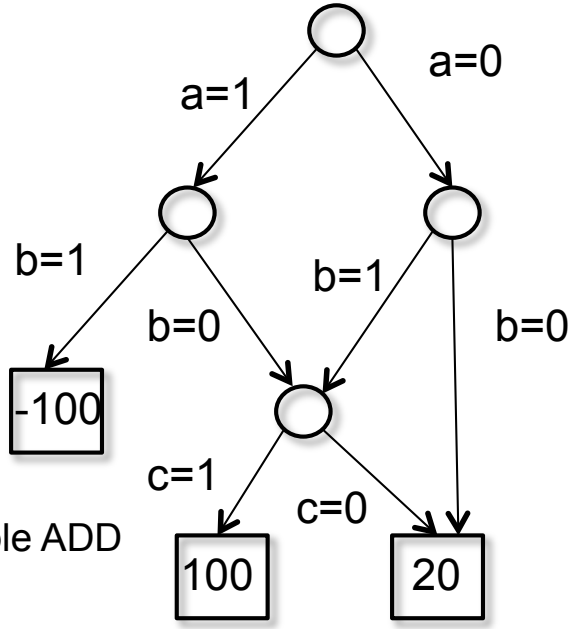


- For reward, let R(s,a,s') be equal to 100 if robot position is Home in s', -100 if robot meets a crocodile in s' and -1 otherwise
- Depends on robot and all crocodiles, function does not factor
- However, has small representation if use tree like this

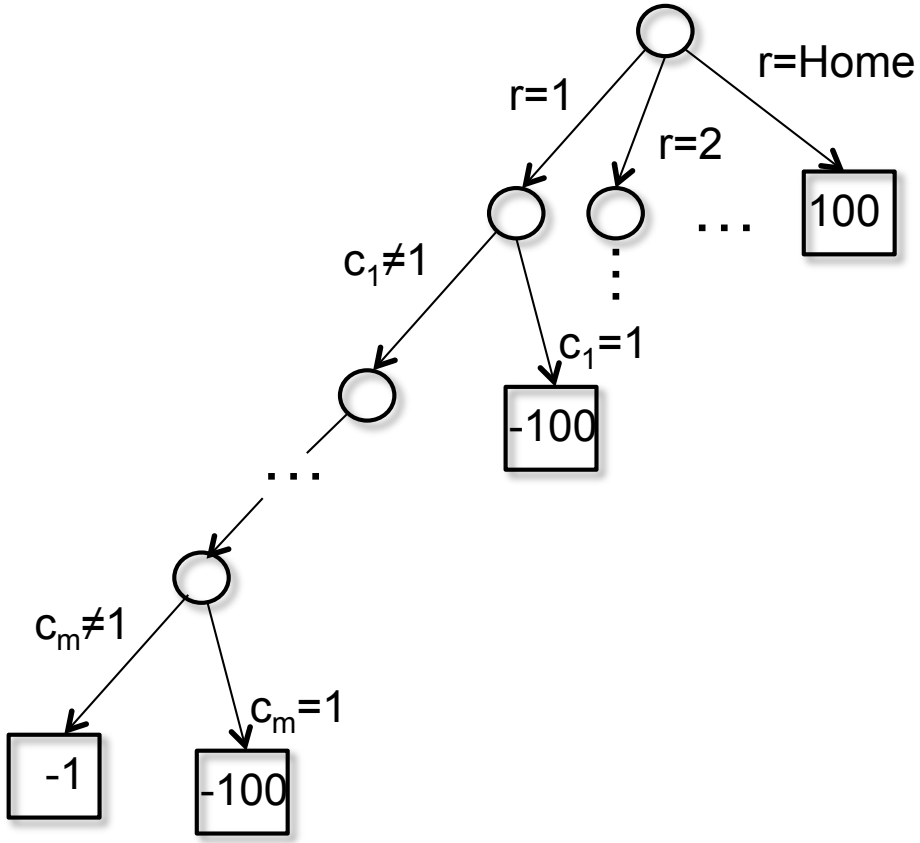


Algebraic Decision Diagram

- Subtrees that are common can be merged, giving a directed acyclic graph
  - When all variables take binary values, called Algebraic Decision Diagram (ADD)



Example ADD



POMDP

Factored POMDP

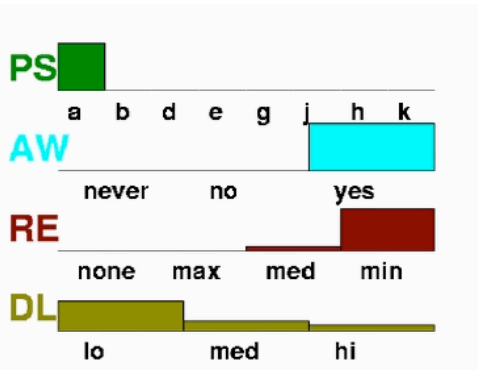
- When state is described by assignment of values to a set of state variables ($s_1, s_2, \ldots, s_m$), the POMDP is often called factored POMDP
- As seen in the robot navigation example, concise description using factored transition functions, ADDs, etc. are often available 
- Unfortunately, even for factored MDP (fully observed), there is no small (poly sized) optimal policy in the worst case 

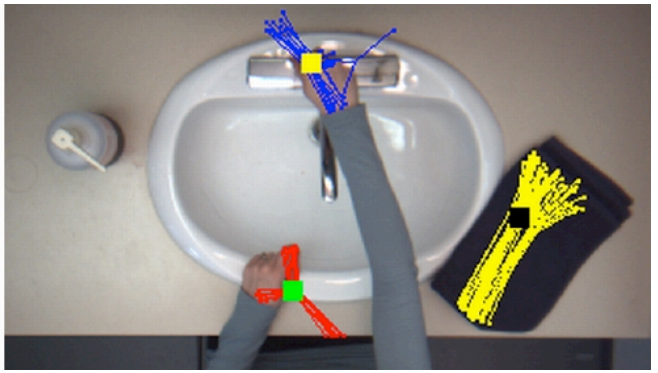
Allender, Arora, Kearns, Moore, Russell 2002

Using Algebraic Decision Diagrams

- Real problems are often not the worst case, so we try …
- One approach is to represent everything using ADDs
  - Represent transition function, observation function and reward function in factored form using ADDs
  - Piecewise constant representation, often concise
  - Represent belief and α-vectors as ADDs
    - Closed under belief updating and α-vector backup operations
    - Representation size may grow with each operation (but hopefully slowly)

Boutilier & Poole, 1996, Hansen & Feng, 2000





- ADD can be combined with point-based method: Symbolic Perseus (Poupart 2005), Symbolic HSVI (Sim, Kim, Kim, Chang, Koo 2008)
- Example success: Assistance for dementia patients

Hoey, Von Bertoldi, Poupart, Mihailidis 2007

Belief Approximation

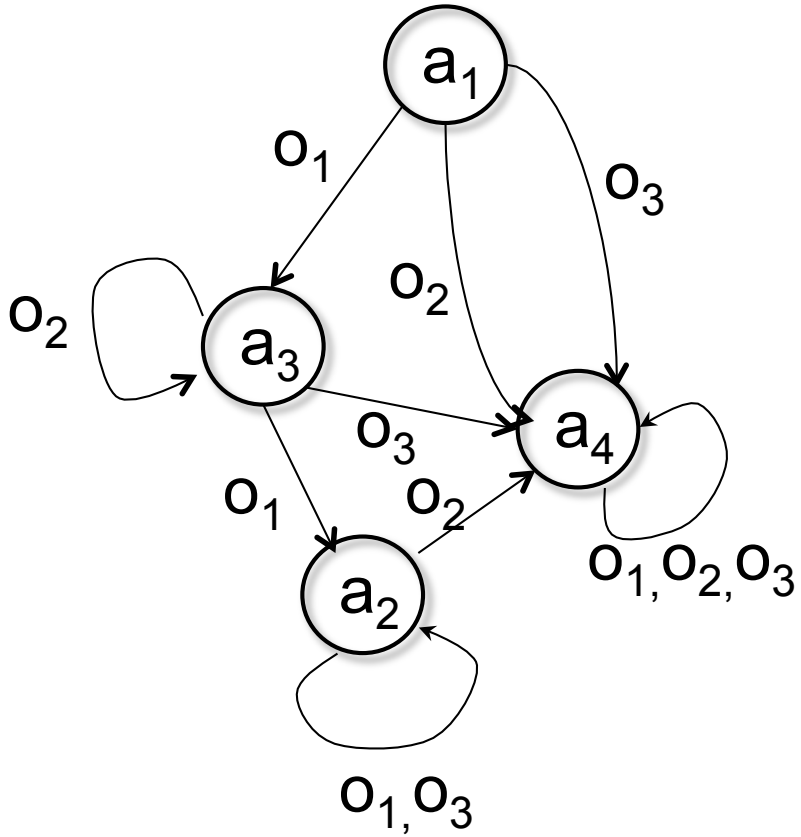
- Flat representation as a vector the same size as state space
  - Finite but practically useless!
- Allow representation to grow with time
  - ADDs can grow (maybe quickly) with each belief update
  - Initial belief b_0 plus history $a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t$ is an exact representation that grows slowly, adds an action and observation at each time step

- From history can do approximate inference
  - Use particle filters (e.g. Thrun 2000, Bai, Hsu, Lee, Ngo 2010)
  - Project onto factored representation (Boyen & Koller 1998, McAllester & Singh 1999)
  - Markov chain Monte Carlo (MCMC) methods (Marthi, Pasula, Russell, Peres 2002)
  - Loopy belief propagation (Murphy, Weiss 2001)
- Difficult but fairly well studied

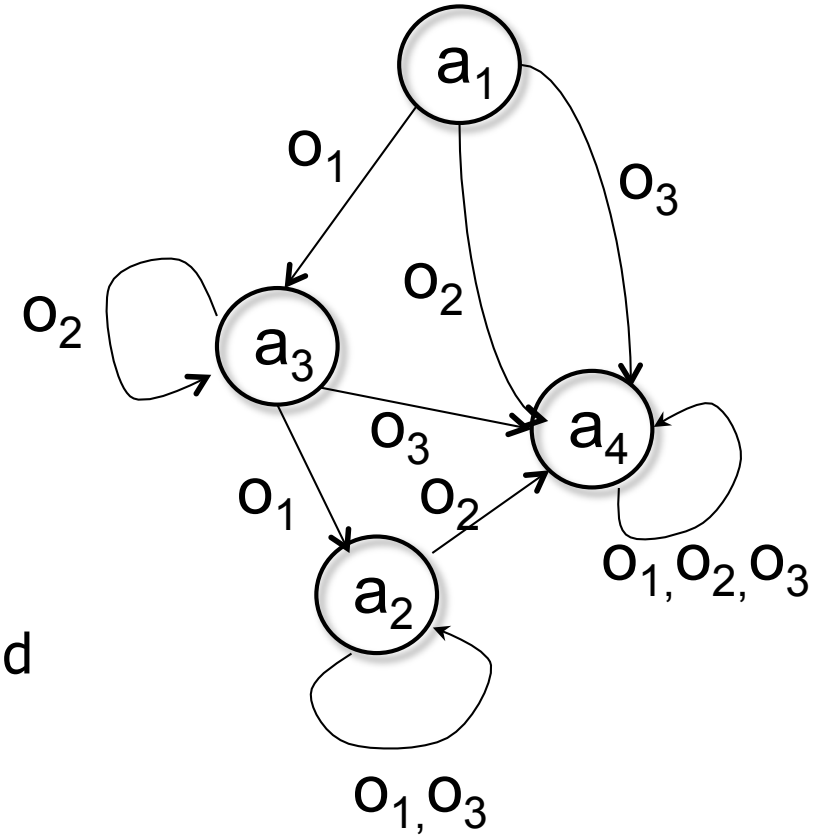
Policy Graph

- Recall that, in point-based methods, we represent value function using a set Γ of α-vectors constructed using backups
  - If a small optimal reachable space exists, then a small number of α-vectors suffices
- Would like to represent α-vectors compactly
  - Flat representation same size as state space
  - ADDs can also grow to be large

- Take policy tree and merge identical subtrees: policy graph
- Policy graph representation requires constant space for each α-vector!
  - But only with approximate rather than exact evaluation of the value $\alpha \cdot b$



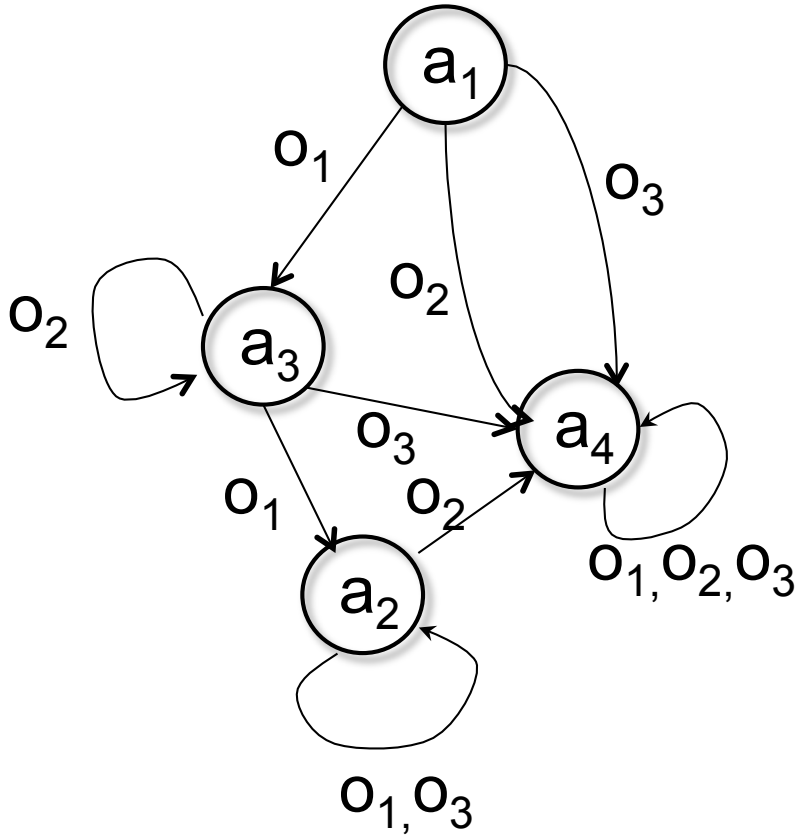
- Policy graph is directed graph with labeled vertices and edges
  - Vertices labeled with action
  - Edges labeled with observation
- To run a policy from a vertex
  - Execute the action at the vertex
  - Follow the resulting observation to the next policy graph node and repeat



- Each policy graph node is associated with an α-vector
- To evaluate

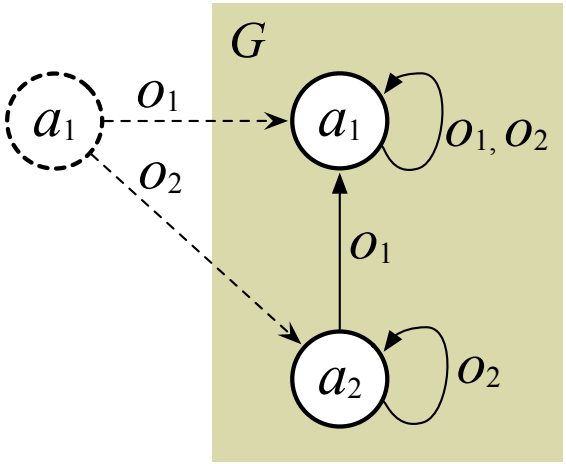
$$\alpha \cdot b = \sum_i \alpha(s_i) b(s_i)$$

  - Monte Carlo method
  - Sample *n* states from *b*
  - Run simulations from the *n* states starting from the policy graph node associated with the α-vector
  - Compute the average simulation rewards
- Evaluation error *O(1/√n)*



Monte Carlo Backup

- Policy graph can be constructed by point-based backup operations
  - Each point-based backup adds one node (one α-vector)
- Conceptually, do Monte Carlo evaluation of all $|A||\Gamma|^{|\Omega|}$ possible ways of adding a new node and choose the one with largest value at b
  - Can be done in time $O(n|A||\Gamma|)$
    - n is the number of simulations used to evaluate a candidate α-vector
  - Selects the candidate with error $O(1/\sqrt{n})$ compared to the best candidate



Details

Bai, Hsu, Lee, Ngo 2010

Monte Carlo Value Iteration

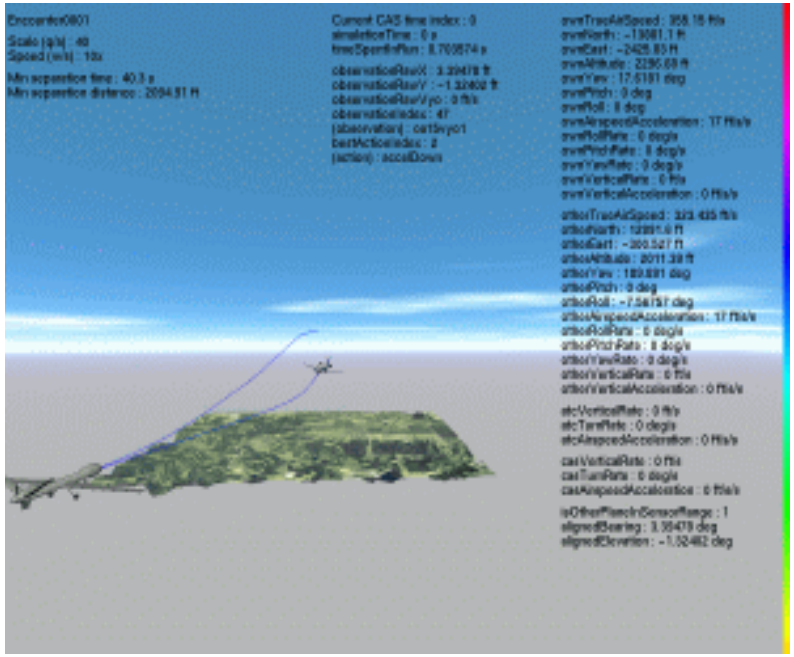
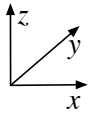
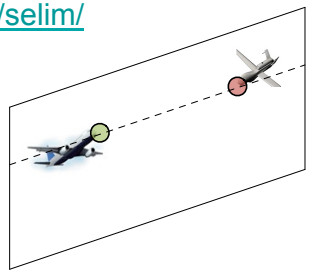


Image from http://web.mit.edu/temizer/www/selim/

	MCVI	TCAS
Risk Ratio	0.0006	0.06





- Use Monte Carlo backup with point-based method to form Monte Carlo Value Iteration
- Application: Aircraft collision avoidance for UAV
  - Particle filters for belief approximation in continuous state space
  - ElectroOptical/Infrared sensor (limited angle)
  - Ability to model large state space allows 3D modeling as opposed to 2D (TCAS)

Bai, Hsu, Lee, Ngo 2010
Bai, Hsu, Kochenderfer, Lee 2011

Outline

- Overview
- Definitions
- Basic properties
- Intractability and Easier Subclasses
- Large state spaces
- Online search

Outline

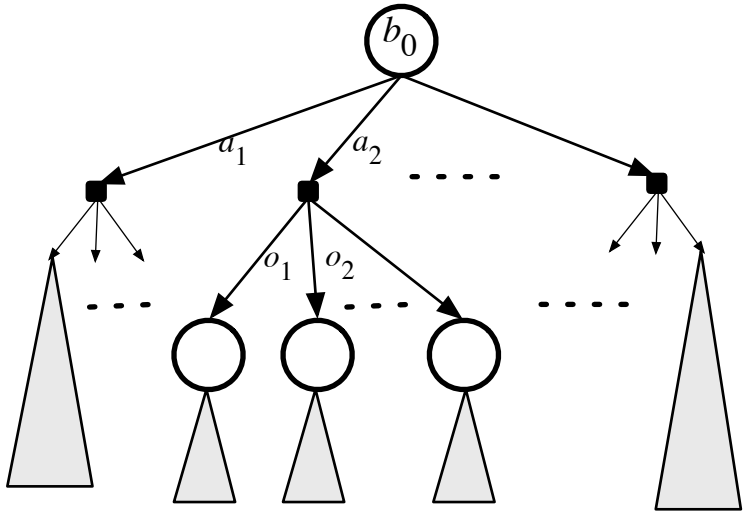
- Online search
  - Branch and bound
  - Large observation space
  - UCT

Online Search

- A policy needs to work well on essentially the whole of an optimal reachable space
  - In the worst case, there is no small policy
- May sometimes still be able to do well, even in this case
  - For example, if a short horizon search is sufficient regardless of current belief
    - Total search space or policy size is large, but only a small amount need to be searched at any one time
- Do online search to look for best current action

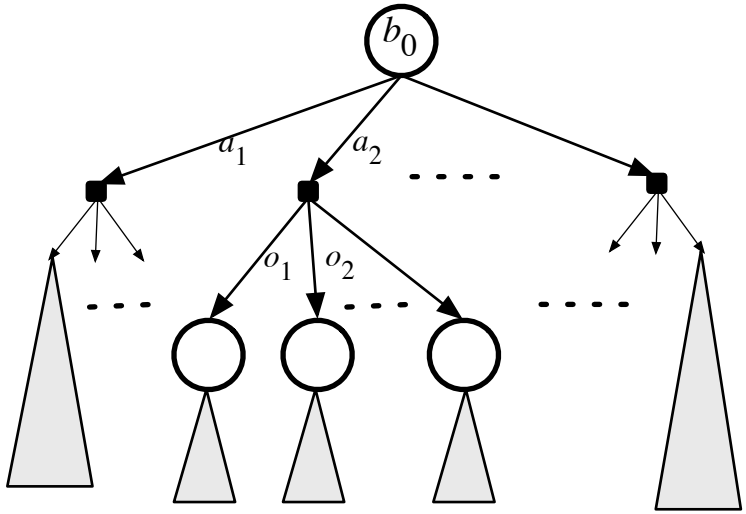
Branch and Bound

- Branch and bound is often used to eliminate parts of search space as part of online search
  - Maintain upper and lower bounds
  - If lower bound is close enough to upper bound to meet target accuracy at root, do not expand node further (whole subtree pruned)



See Ross, Pineau, Paquet, Chaib-draa 2008 for survey

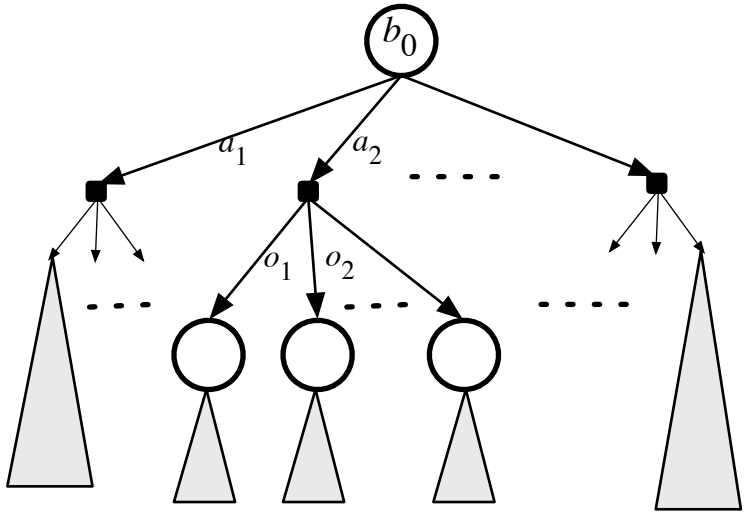
- Heuristics used to order nodes for expansion, e.g. from HSVI (Smith & Simmons 2004),
  - Choose action with the largest upper bound
  - Observation that contributes the most to accuracy at root
- Target accuracy at root is continuously strengthened while time remains, resulting in deeper and deeper search



Online search is essentially the same as offline policy search, except that values are back up to the root without generating α-vectors

Large Observation Space

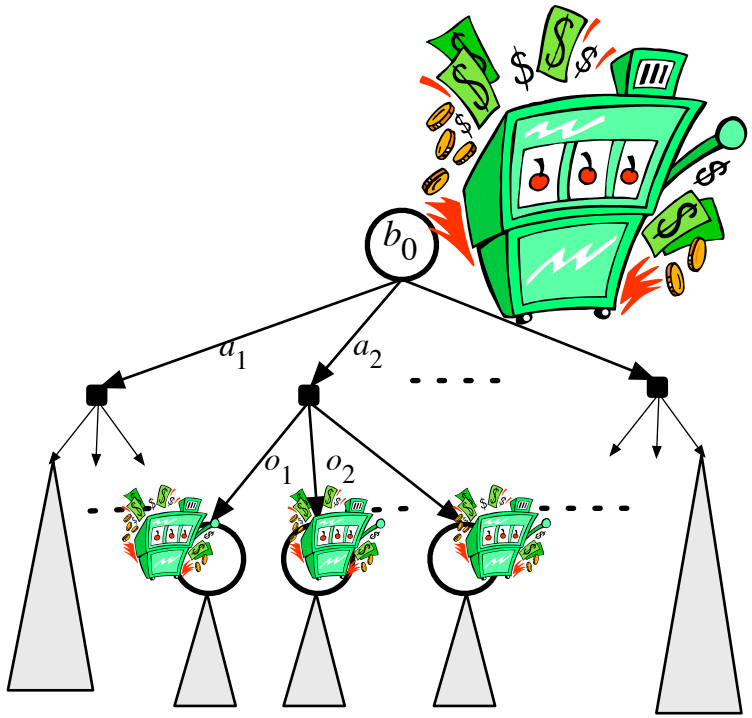
- With horizon k, search space size is $O(|A|^k|\Omega|^k)$
  - Large when action or observation space is large
- Can sample observations to get $O(|A|^k|poly(1/\varepsilon)|^k)$ for accuracy ε
  - Very large observation space okay
  - Very large reachable belief space okay if action space small and horizon short



Kearns, Mansour, Ng 1999
McAllester & Singh 1999

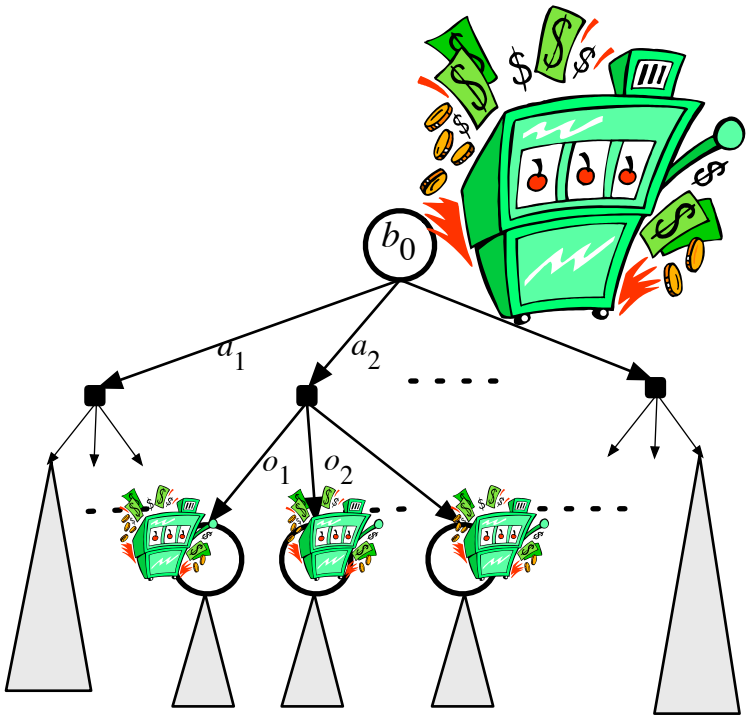
Upper Confidence Tree

- Branch and bound requires upper bound on value function
  - Bound need to be optimistic for correctness
- UCT use multi-arm bandit algorithm at each node
  - Maintain upper confidence interval for each action at each belief
  - Repeatedly start simulation from root
    - Select action according to upper bound, observation randomly to simulate path down the tree
    - Update upper bound using value backed up from the path



Kocsis & Szepesvari 2006
Silver & Veness 2010

- Let
  - N_b : number of times the belief has been encountered
  - $N_{b,a}$: number of times action a taken at b
  - $V_{b,a}$: current average value of taking action a at b
- For upper confidence interval, use $V_{b,a} + c\sqrt{\frac{\log N_b}{N_{b,a}}}$
  - Can be shown to converge for appropriate value of constant c



Computer Go

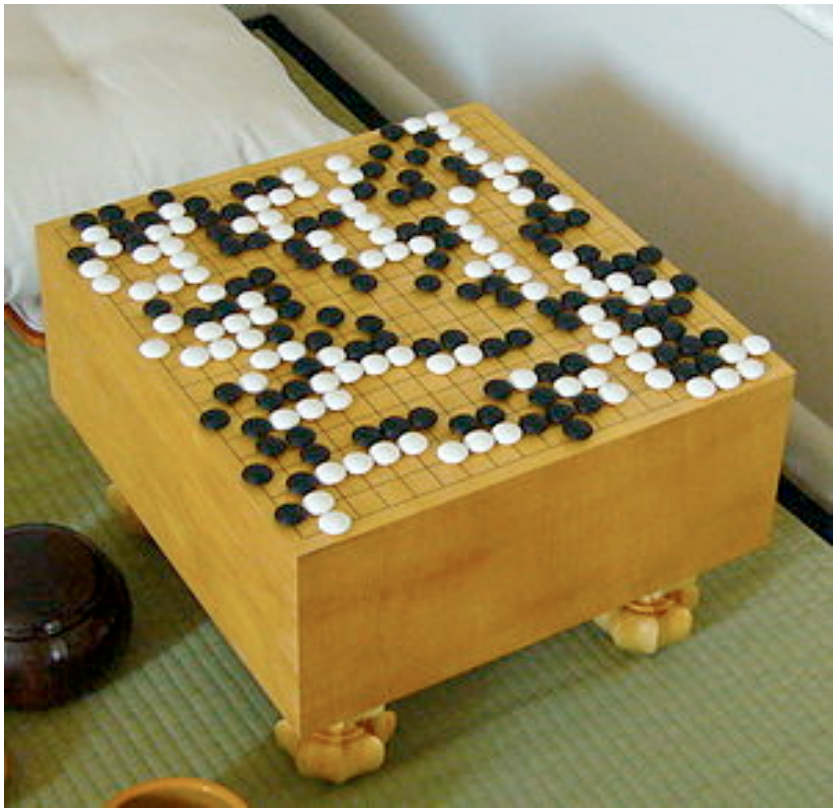


Image from http://en.wikipedia.org/wiki/Go_%28board_game%29

- UCT very successful in related area of game tree search, particularly in computer Go
- Ing Prize of 40 million NT dollars for beating a 1-dan professional human player unclaimed in 2000
  - Computers were hopeless then
- Around 2006, UCT introduced and used for Go
  - Very rapid improvement in strength
- Now high dan level on 9 x 9 board and low dan level on full 19 x 19 board

Review

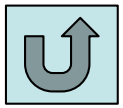
- Overview
- Definitions
- Basic properties
- Intractability and Easier Subclasses
- Large state spaces
- Online search

Appendix

- Belief Update Equation
- Piecewise linear value function
- Convergence and Bellman Optimality Equation
- Efficient approximation with small belief space
- MC Backup
- References

Belief Update Equation

$$
\begin{aligned}
b_t(s_t) &= \Pr(s_t|a_0,\ldots,a_{t-1},o_t) \\
&= \frac{\Pr(o_t|a_0,\ldots,a_{t-1},s_t)\Pr(s_t|a_0,\ldots,a_{t-1})}{\Pr(o_t|a_0,\ldots,a_{t-1})} \\
&\quad \text{using Bayes rule} \\
&= \frac{O(o_t,a_{t-1},s_t)\sum_{s_{t-1}}\Pr(s_t,s_{t-1}|a_0,\ldots,a_{t-1})}{\Pr(o_t|a_0,\ldots,a_{t-1})} \\
&\quad o_t \text{ independent of history given } s_t \text{ and } a_{t-1}\text{, marginalization} \\
&= \frac{O(o_t,a_{t-1},s_t)\sum_{s_{t-1}}\Pr(s_t|a_0,\ldots,a_{t-1},s_{t-1})\Pr(s_{t-1}|a_0,\ldots,o_{t-1},a_{t-1})}{\Pr(o_t|a_0,\ldots,a_{t-1})} \\
&\quad \text{using chain rule} \\
&= \frac{O(o_t,a_{t-1},s_t)\sum_{s_{t-1}}T(s_{t-1},a_{t-1},s_t)b_{t-1}(s_{t-1})}{\Pr(o_t|a_0,\ldots,a_{t-1})} \\
&\quad s_{t-1} \text{ independent of } a_{t-1}
\end{aligned}
$$



Piecewise Linear Value Function

- When horizon is one, value function is

$$
\begin{aligned}
V_1(b) &= \max_a R(a,b) \\
&= \max_a \sum_i \sum_j T(s_i,a,s_j) b(s_i) R(s_i,a,s_j) \\
&= \max_a \sum_i b(s_i) \left(\sum_j T(s_i,a,s_j) R(s_i,a,s_j) \right) \\
&= \max_a \sum_i \alpha(s_i) b(s_i)
\end{aligned}
$$

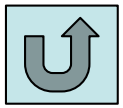
  - Maximum of |A| linear function
    - Value function is piecewise linear

NUS National University of Singapore

- Assume as inductive hypothesis that $V^*_{i-1}(b) = \max_{\alpha \in \Gamma_{i-1}} \alpha \cdot b$

$$
\begin{aligned}
V^*_i(b) &= \max_a R(a,b) + \gamma \sum_o \Pr(o|a,b) V^*_{i-1}(\tau(o,a,b)) \\
&= \max_a R(a,b) + \gamma \sum_o \Pr(o|a,b) \max_{\alpha_o \in \Gamma_{i-1}} \alpha_o \cdot \tau(o,a,b) \\
&= \max_a \sum_s \sum_{s'} T(s,a,s') R(s,a,s') b(s) \\
&\quad + \gamma \sum_o \Pr(o|a,b) \max_{\alpha_o \in \Gamma_{i-1}} \sum_s \alpha_o(s) \frac{O(o,a,s) \sum_{s'} T(s,a,s') b(s)}{\Pr(o|a,b)} \\
&= \max_a \max_{\alpha_{o_1} \in \Gamma_{i-1}, \ldots, \alpha_{o_{|\Omega|}} \in \Gamma_{i-1}} \sum_s \sum_{s'} T(s,a,s') R(s,a,s') b(s) \\
&\quad + \gamma \sum_{o \in \{o_1, \ldots, o_{|\Omega|}\}} \sum_s \alpha_o(s) O(o,a,s) \sum_{s'} T(s,a,s') b(s) \\
&= \max_{\alpha \in \Gamma_i} \alpha \cdot b
\end{aligned}
$$

POMDP



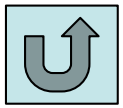
Convergence and Bellman Optimality Equation

- Contraction $||HU - HV||_\infty \leq \gamma ||U - V||_\infty$
  - Without loss of generality, assume HU(b)≥H(V(b)
  - Let a* be optimal for HU(b)

$$
\begin{aligned}
HU(b) - HV(b) &\leq R(a^*, b) + \gamma \sum_o p(o|a^*, b) U(\tau(o, a, b)) \\
&\quad - R(a^*, b) - \gamma \sum_o p(o|a^*, b) V(\tau(o, a, b)) \\
&= \gamma \sum_o p(o|a^*, b)[U(\tau(o, a, b)) - V(\tau(o, a, b))] \\
&\leq \gamma \sum_o p(o|a^*, b) ||U - V||_\infty \\
&= \gamma ||U - V||_\infty
\end{aligned}
$$

NUS National University of Singapore

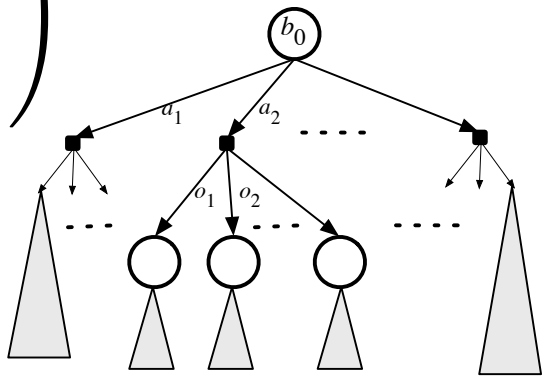
- *H* is a contractive mapping
  - By the Banach fixed point theorem, value iteration $V_i = HV_{i-1}$ converges to a unique fixed point V^* such that $V^* = HV^*$



Efficient Approximation with Small Belief Space

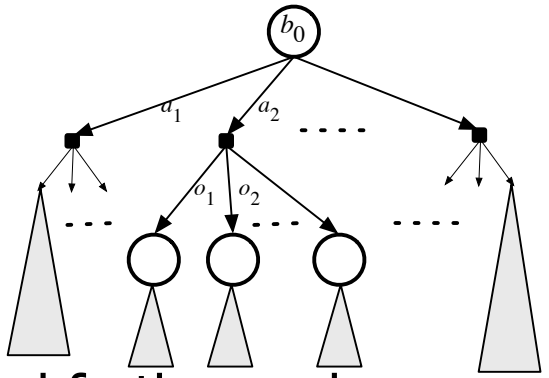
- Policy with $|V^*(b_0) - V(b_0)| \leq \epsilon$ can be found in time

$$O\left(\mathcal{C}\left(\frac{(1-\gamma)^2\epsilon}{4\gamma R_{\max}}\right)^2 \log_\gamma \frac{(1-\gamma)\epsilon}{2R_{\max}}\right)$$



- Proof Sketch:
  - Search belief tree from b_0
  - Discounting means searching tree of height O(log ε) is sufficient

– Do a depth first search, backup *α -vector* at a node after searching children



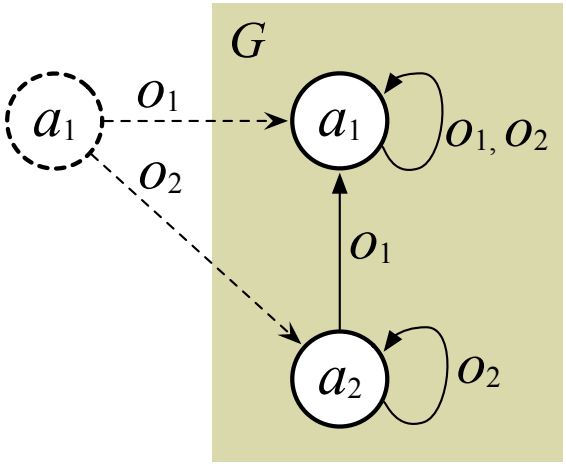
- Maintain a set of beliefs C_i at height i
- If newly discovered belief within distance δ to a belief in C_i do not expand further and use policy of nearest belief in C_i (approximate dynamic programming)
- By appropriately setting δ, can show that the size of C_i is at most

$$\mathcal{C}\left(\frac{(1-\gamma)^2\epsilon}{4\gamma R_{\max}}\right)$$

- Runtime is time to find nearest neighbours within C_i

MC Backup

- For each action *a* in *A*
  - Sample *n* states from *b*. For each sample *i*
    - Generate s_i' from $T(s_i,a,s_i')$ and o_i from $O(o_i,a,s_i')$, record reward
  - For each *o* in Ω
    - For each α in Γ and all s_i' associated with *o*
      - Run simulation from s_i' and α, record return
    - Find best α and associate it with *o*
  - Average the return from the *n* simulations from *b* associated with the best α's to get score for *a*
- Find the best *a*
- Construct the new policy node using the best *a* and it's associated α's



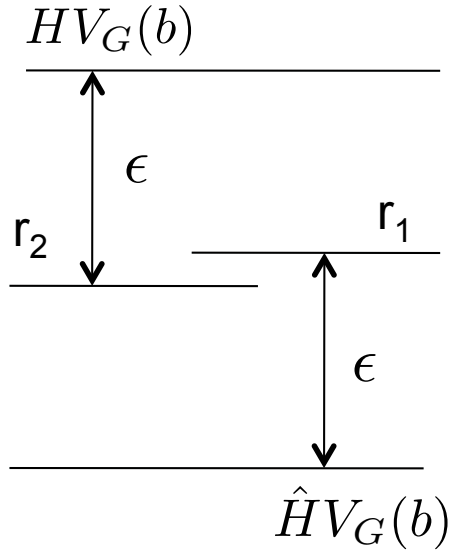
NUS National University of Singapore

- Let G be the current policy graph.
  Let value of point-based backup at *b* be $HV_G(b)$
  value of MC-backup be $\hat{H}V_G(b)$
  With probability at least 1- τ

$$|HV_G(b) - \hat{H}_b V_G(b)| \leq \frac{2R_{\max}}{1-\gamma}\sqrt{\frac{2(|O|\ln|G| + \ln(2|A|) + \ln(1/\tau))}{n}}$$

- Proof sketch:
  There are |A||G|$^{|\Omega|}$ possible new graphs.
  Let C be $R_{max}/(1-\gamma)$.
  For a single graph, Hoeffding inequality shows that the probability that the average return is further than ε from the expected return is no more than $2e^{-n\epsilon^2/2C^2}$

- Using the union bound, the probability that any of the possible graphs does not satisfy the same condition is no more than $|A||G|^{\Omega}2e^{-n\epsilon^2/2C^2}$ Set this value to τ .
- The graph with the best average return r_1 is selected
  - It's expected return is at most ε lower
  - It's average return is higher than average return r_2 of HV_G, whose expected return is at most ε higher.
  - Hence, difference in expected return at most 2ε
- Manipulating the expressions gives the required bounds



References

- E. Allender, S. Arora, M. Kearns, C. Moore, and A. Russell. A note on the representational incompatibility of function approximation and factored dynamics. Advances in Neural Information Processing Systems, pages 447– 454, 2003.
- H. Bai, D. Hsu, M.J. Kochenderfer, and W. S. Lee. Unmanned aircraft collision avoidance using continuous-state POMDPs. In Proc. Robotics: Science & Systems, 2011.
- H. Bai, D. Hsu, W. S. Lee, and V. Ngo. Monte Carlo Value Iteration for Continuous-State POMDPs. In Proc. Int. Workshop on the Algorithmic Foundations of Robotics (WAFR), pages 175–191. Springer, 2011.
- C. Boutilier and D. Poole. Computing optimal policies for partially observ- able decision processes using compact representations. In Proceedings of the National Conference on Artificial Intelligence, pages 1168–1175, 1996.
- X. Boyen and D. Koller. Tractable inference for complex stochastic pro- cesses. In Proc. UAI, volume 98, 1998.

- E.A. Hansen and Z. Feng. Dynamic programming for POMDPs using a factored state representation. In Proceedings of the Fifth International Conference on AI Planning Systems, pages 130–139, 2000.
- J. Hoey, A. Von Bertoldi, P. Poupart, and A. Mihailidis. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In Proc. Int. Conf. on Vision Systems, volume 65, page 66, 2007.
- D. Hsu, W.S. Lee, and N. Rong. What makes some POMDP problems easy to approximate. Advances in Neural Information Processing Systems (NIPS), 2007.
- M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2, pages 1324–1331. Morgan Kaufmann Publishers Inc., 1999.
- L.L. Ko, D. Hsu, W.S. Lee, and S.C.W. Ong. Structured parameter elicitation. In Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.

- MJ Kochenderfer and JP Chryssanthacopoulos. Robust airborne collision avoidance through dynamic programming. Massachusetts Institute of Tech- nology, Lincoln Laboratory, Project Report ATC-371, 2011.
- L. Kocsis and C. Szepesvari. Bandit based monte-carlo planning. Machine Learning: ECML 2006, pages 282–293, 2006.
- H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In Proc. Robotics: Science & Systems, 2008.
- M.L. Littman, A.R. Cassandra, and L.P. Kaelbling. Learning policies for partially observable environments: Scaling up. In International Conference on Machine Learning, pages 362–370, 1995.
- B. Marthi, H. Pasula, S. Russell, and Y. Peres. Decayed mcmc filtering. In Proceedings of UAI, 2002.

- D. McAllester and S. Singh. Approximate planning for factored POMDPs using belief state simplification. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pages 409–416, 1999.
- K. Murphy and Y. Weiss. The factored frontier algorithm for approximate inference in dbns. In Proc. of the Conf. on Uncertainty in AI, 2001.
- M.D. Ngo. Scaling techniques for intelligent assistants in collaborative games. B.Comp. Dissertation, School of Computing, National University of Singapore, 2011.
- S.C.W. Ong et al. Planning under uncertainty for robotic tasks with mixed observability. The International Journal of Robotics Research, 29(8):1053, 2010.
- C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of Markov decision processes. Mathematics of operations research, pages 441–450, 1987.

- J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An any- time algorithm for POMDPs. In Int. Jnt. Conf. on Artificial Intelligence, volume 18, pages 1025–1032, 2003.
- P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In Proceedings of the 23rd inter- national conference on Machine learning, pages 697–704, 2006.
- S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. Online planning algorithms for pomdps. Journal of Artificial Intelligence Research, 32(1): 663– 704, 2008.
- N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. Journal of Artificial Intelligence Research, 23(1):1–40, 2005.
- D. Silver and J. Veness. Monte-carlo planning in large POMDPs. Advances in Neural Information Processing Systems (NIPS), 2010.

- R.D. Smallwood and E.J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. Operations Research, 21(5):1071–1088, 1973.
- T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In Proc. Conf. on Uncertainty in Artificial Intelligence, pages 520–527. AUAI Press, 2004.
- T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In Proc. Uncertainty in Artificial Intelligence, 2005.
- S. Temizer, M.J. Kochenderfer, L.P. Kaelbling, T. Lozano-Perez, and J.K. Kuchar. Collision avoidance for unmanned aircraft using Markov deci- sion processes. In AIAA Guidance, Navigation, and Control Conference, Toronto, Canada, 2010.
- J.D. Williams and S. Young. Partially observable Markov decision processes for spoken dialog systems. Computer Speech & Language, 21(2):393–422, 2007.