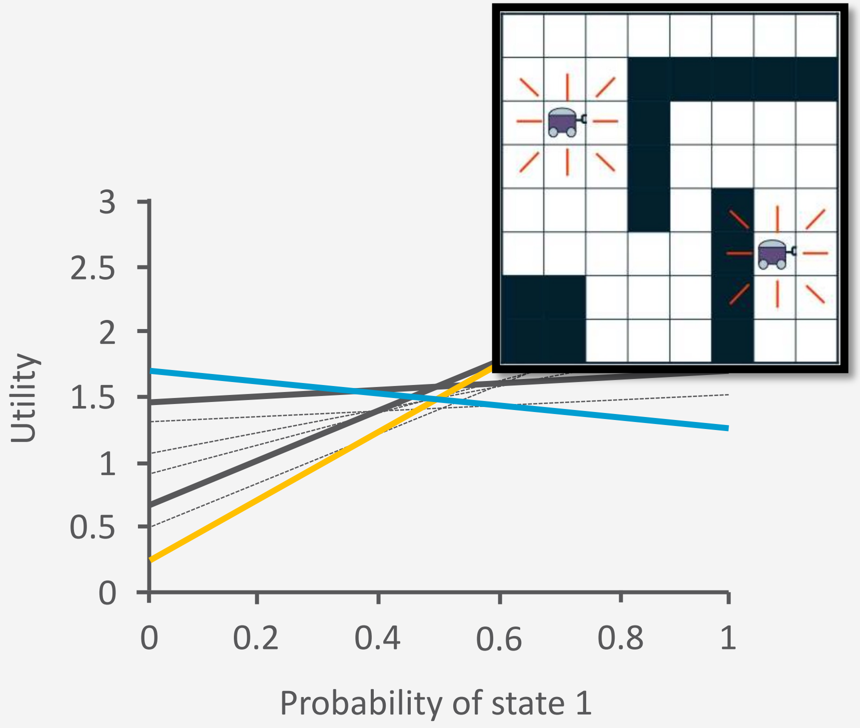
Universität Münster



Automated Planning and Acting

Decision Making: Extensions

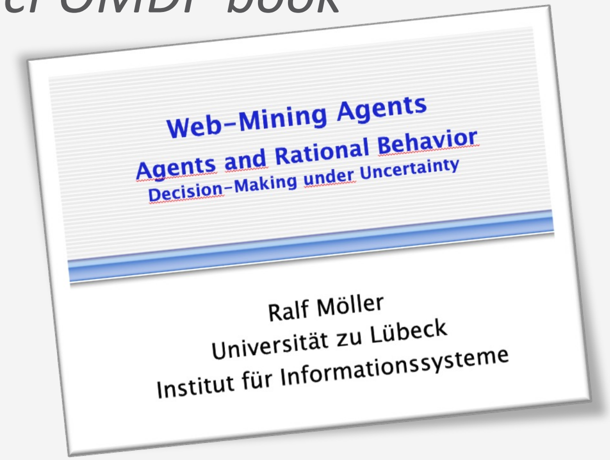
living.knowledge

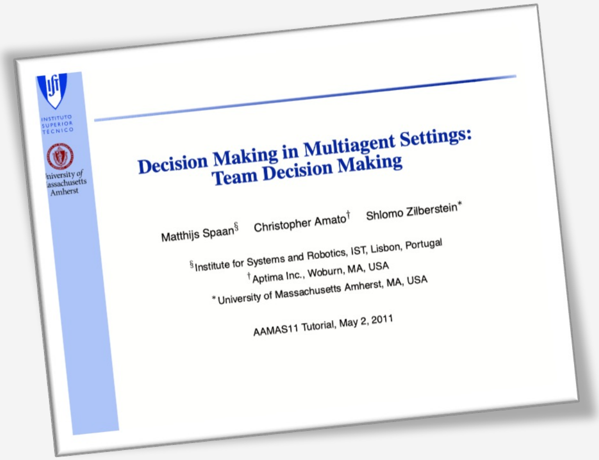
Tanya Braun
Research Group Data Science, Computer Science Department

Content: Planning and Acting

Acknowledgements

- Part 1 based on material from Lise Getoor, Jean-Claude Latombe, Daphne Koller, and Stuart Russell, Xiaoli Fern compiled by Ralf Möller
  - In part based on *AIMA Book, Chapter 17.4*
- Part 2 based on tutorial by Matthijs Spaan, Christopher Amato, Shlomo Zilberstein on *Decision Making in Multiagent Settings: Team Decision Making*
  - In part based on *DecPOMDP book*





http://rbr.cs.umass.edu/camato/decpomdp/overview.html

Outline: Decision Making – Extensions

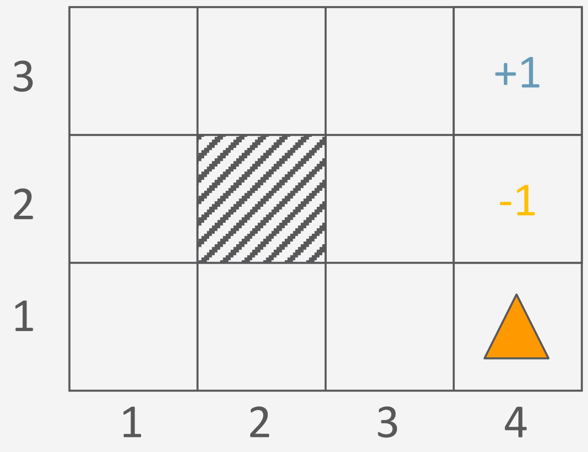
Partially Observable Markov Decision Process (POMDP)

- POMDP agent, belief state, belief MDP
- Conditional plans, value iteration

Decentralised POMDP (Dec-POMDP)

- Dec-POMDP, local policy, joint policy, value function
- Communication, full observability, Dec-MDP
- Solutions for finite, infinite, indefinite horizon

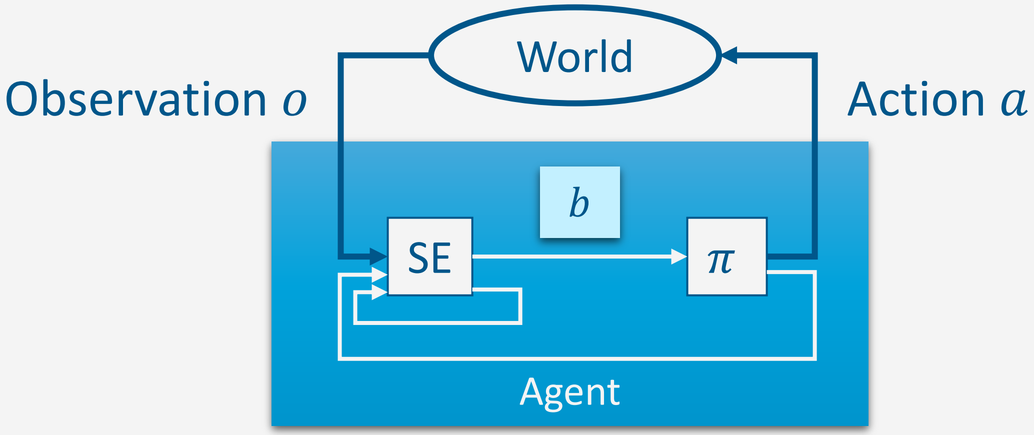
POMDP



- POMDP = Partially Observable MDP
  - Sensing operation returns multiple states, with a probability distribution
    - Sensor model Ω that encodes $P(o|s)$ (or $P(o|s,a)$)
      - Probability of observing o given state s (and action a)
    - Example:
      - Sensing number of adjacent walls (1 or 2)
      - Return correct value with probability 0.9
  - Formally, POMDP is a six-tuple (S, A, T, R, O, Ω)
    - MDP (S, A, T, R) extended with a set of observations O and a sensor model Ω
  - Choosing action that maximizes expected utility of state distribution assuming "state utilities" computed as before not good enough → Does not make sense (not rational)
- POMDP agent: Constructing a new MDP in which the current probability distribution over states plays the role of the state variable

Decision cycle of a POMDP agent

- Given the current belief state b and a policy π, execute the action
$$a = \pi(b)$$
- Receive observation o
- Set the current belief state to $SE(b, a, o)$ and repeat
  - SE = State Estimation



Belief State & Update

- Belief state $b(s)$ is the probability assigned to the actual state s by belief state b
- Initial belief state
  - Probability of 0 for terminal states
  - Uniform distribution for rest
  - Robot navigation example:
    - $b = \left(\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0,0\right)$

3	$0.\overline{1}$	$0.\overline{1}$	$0.\overline{1}$	0.0
2	$0.\overline{1}$		$0.\overline{1}$	0.0
1	$0.\overline{1}$	$0.\overline{1}$	$0.\overline{1}$	$0.\overline{1}$
	1	2	3	4

Belief State & Update

- Update $b' = SE(b, a, o)$

$$b'(s') = P(s'|o,a,b) = \frac{P(o|s',a)\sum_{s\in \mathrm{dom}(S)} P(s'|s,a)b(s)}{\sum_{s''\in \mathrm{dom}(S)} P(o|s'',a)\sum_{s\in \mathrm{dom}(S)} P(s''|s,a)b(s)}$$

  - Consider as two-stage update: (1) Update for the action (2) Update for the observation

b

	1	2	3	4
3	$0.\bar{1}$	$0.\bar{1}$	$0.\bar{1}$	0.0
2	$0.\bar{1}$		$0.\bar{1}$	0.0
1	$0.\bar{1}$	$0.\bar{1}$	$0.\bar{1}$	$0.\bar{1}$

Move L once

$b^{(1)}$

	1	2	3	4
3	0.2	$0.\bar{1}$	$0.0\bar{2}$	0.0
2	$0.\bar{1}$		$0.\bar{1}$	$0.0\bar{1}$
1	0.2	$0.\bar{1}$	$0.\bar{1}$	$0.0\bar{1}$

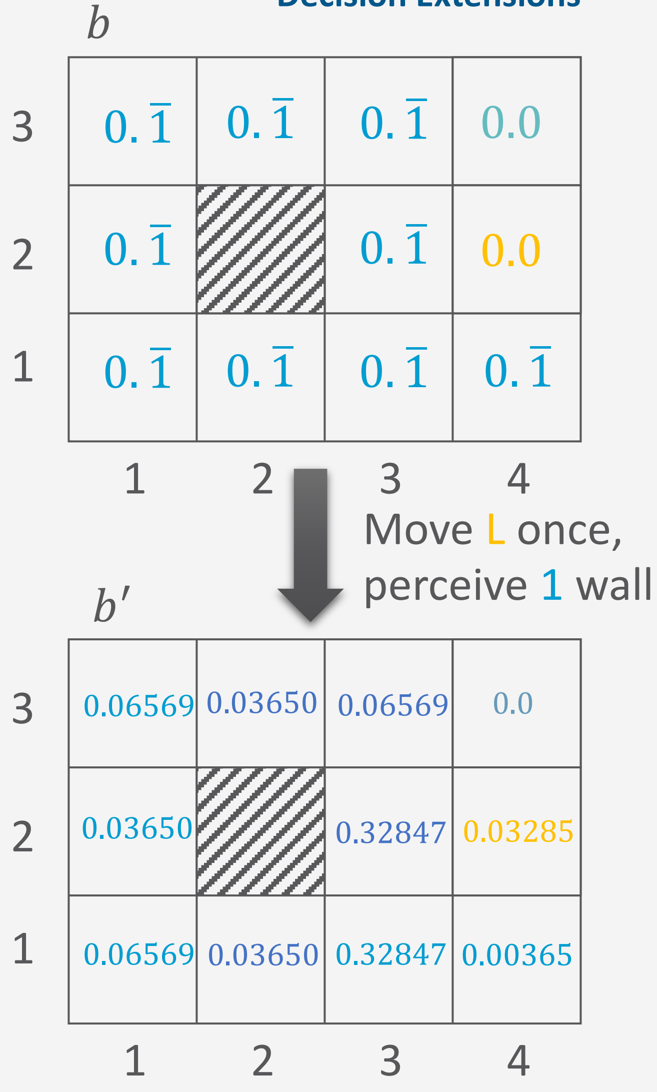
Perceive 1 wall

$b^{(2)} = b'$

	1	2	3	4
3	0.06569	0.03650	0.06569	0.0
2	0.03650		0.32847	0.03285
1	0.06569	0.03650	0.32847	0.00365

Belief MDP

- A belief MDP is a tuple $(B, A, \rho, T, O, \Omega)$
  - B = *infinite* set of belief states
    - Continuous!
  - A = finite set of actions
  - Reward function $\rho(b)$ (can also be defined with a)
    - Reward of belief state b
  - Transition function $T(b', b, a) = P(b'|b, a)$
    - Probability of new belief state b' given belief state b and action a
  - O = finite set of observations
  - Sensor model $\Omega(o, b) = P(o|b)$ (can also be defined with a)
    - Probability of observation o given belief state b (and action a)



Belief MDP: Express Functions using POMDP Functions

- Reward function: Sum over all actual states that the agent can be in

$$\rho(b) = \sum_{s} b(s)R(s)$$

- Transition function: Sum over all possible observations

$$P(b'|b,a) = \sum_{o} P(b'|o,a,b)P(o|a,b) = \sum_{o} P(b'|o,a,b) \sum_{s'} P(o|s') \sum_{s} P(s'|s,a)b(s)$$

  - where $P(b'|o,a,b) = 1$ if $b' = SE(b,a,o)$ and 0 oth.
- Sensor model: Sum over all actual states that the agent might reach

$$P(o|a,b) = \sum_{s'} P(o|a,s',b)P(s'|a,b) = \sum_{s'} P(o|s')P(s'|a,b) = \sum_{s'} P(o|s') \sum_{s} P(s'|s,a)b(s)$$

- $P(b'|b,a)$ and $\rho(b)$ define an observable MDP on the space of belief states

Belief MDP

- Optimal action depends only on agent’s current belief state
  - Does not depend on actual state the agent is in

$\Rightarrow$ Solving a POMDP on a physical state space is reduced to solving an MDP on the corresponding belief-state space
  - Mapping $\pi^*(b)$ from belief states to actions

b

3	$0.\bar{1}$	$0.\bar{1}$	$0.\bar{1}$	0.0
2	$0.\bar{1}$		$0.\bar{1}$	0.0
1	$0.\bar{1}$	$0.\bar{1}$	$0.\bar{1}$	$0.\bar{1}$
	1	2	3	4

Move L once, perceive 1 wall

b'

3	0.06569	0.03650	0.06569	0.0
2	0.03650		0.32847	0.03285
1	0.06569	0.03650	0.32847	0.00365
	1	2	3	4

Example Scenario

	1	2	3	4
3	0.622	0.221	0.071	0.024
2	0.005		0.003	0.022
1	0.003	0.024	0.003	0.000

After moving U five times

	1	2	3	4
3	0.300	0.010	0.008	0.000
2	0.221		0.059	0.012
1	0.371	0.012	0.008	0.000

After moving L five times

	1	2	3	4
3	0.111	0.111	0.111	0.000
2	0.111		0.111	0.000
1	0.111	0.111	0.111	0.111

Initial distribution

	1	2	3	4
3	0.005	0.007	0.019	0.775
2	0.034		0.007	0.105
1	0.005	0.006	0.008	0.030

After moving R five times

Conditional Plans

- Example:
  - Two state world 0,1
  - Two actions: $stay(P), go(P)$
    - Actions achieve intended effect with some probability P
  - One-step plan $[go], [stay]$
- Two-step plans are conditional
  - $[a1,$ IF $percept = 0$ THEN $a2$ ELSE $a3]$
  - Shorthand notation: $[a1, a2/a3]$
- n-step plans are trees with
  - Nodes attached with actions and
  - Edges attached with percepts

Value Iteration for POMDPs

- Cannot compute a single utility value for each state of all belief states
- Consider an optimal policy π^* and its application in belief state b
- For this b, the policy is a conditional plan p
  - Let the utility of executing a fixed conditional plan p in s be $u_p(s)$
  - Expected utility $U_p(b) = \sum_s b(s)u_p(s)$
    - It varies linearly with b, a hyperplane in a belief space
  - At any b, the optimal policy will choose the conditional plan with the highest expected utility

$$U(b) = U^{\pi^*}(b) = \max_p \sum_s b(s)u_p(s)$$

$$\pi^* = \arg\max_p \sum_s b(s)u_p(s)$$

    - $U(b)$ is the maximum of a collection of hyperplanes and will be piecewise linear and convex

General Formula

- Let p be a depth-d conditional plan whose initial action is a and whose depth-$d-1$ subplan for percept e is $p.e$, then

$$u_p(s) = R(s) + \sum_{s'} P(s' \mid s, a) \sum_e P(e|s')\, u_{p.e}(s')$$

  - $d = 0$: $u_p(s) = R(s)$ for the empty plan $p = \perp$
  - $d = 1$: $p.e = \perp$ for all e, simplifying the last sum:

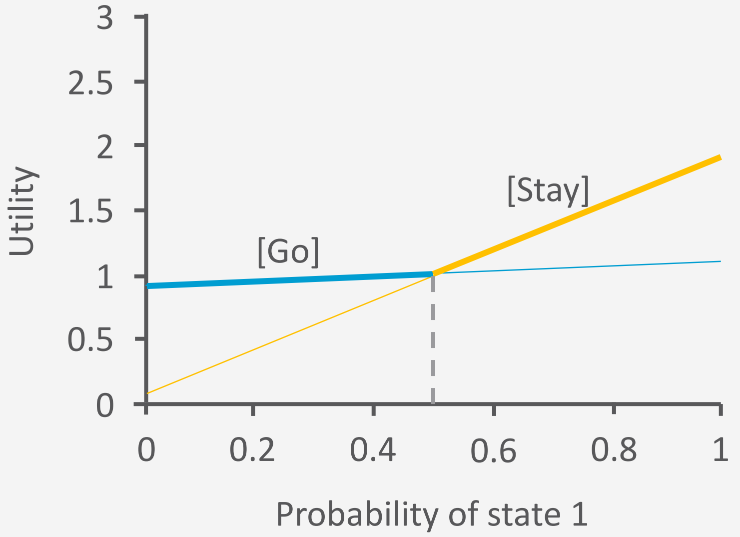
$$\sum_e P(e|s')\, u_{p.e}(s') = \sum_e P(e|s')\, u_\perp(s') = u_\perp(s') \sum_e P(e|s') = u_\perp(s') \cdot 1 = R(s')$$

- This gives us a *value iteration* algorithm
  - Elimination of dominated plans is essential for reducing doubly exponential growth:
    - Number of undominated plans with $d = 8$ is just 144, otherwise 2^{255} $(|A|^{O\left(|E|^{d-1}\right)})$
    - For large POMDPs this approach is highly inefficient

Example

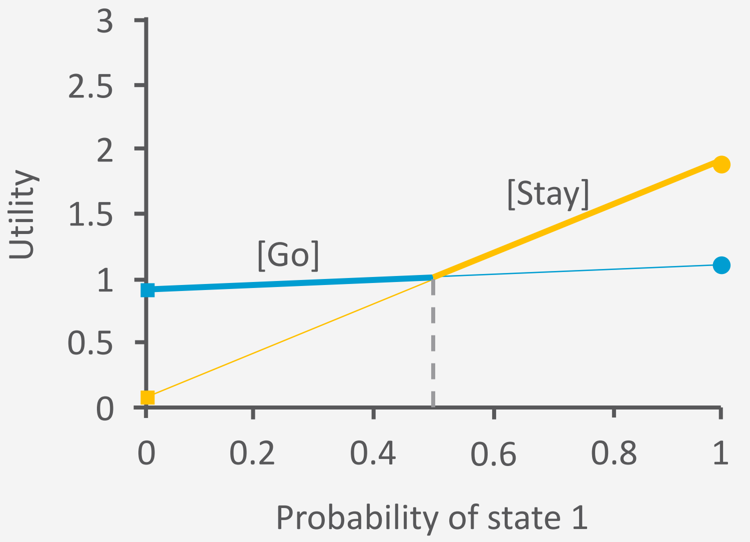
- Compute the utilities for conditional plans of depth 2 by
  - considering each possible first action
  - each possible subsequent percept
  - each way of choosing a depth-1 plan to execute for each percept

Utility of two one-step plans as a function of $b(1)$



Example

- Two state world 0,1
- Rewards $R(0) = 0, R(1) = 1$
- Two actions: $stay(0.9), go(0.9)$
- Sensor reports correct state with probability of 0.6
- Consider the one-step plans $[stay]$ and $[go]$
  - $u_{[stay]}(0) = R(0) + \underbrace{0.9R(0)}_{\text{state 0}} + \underbrace{0.1R(1)}_{\text{state 1}} = 0.1$
  - $u_{[stay]}(1) = R(1) + 0.1R(0) + 0.9R(1) = 1.9$
  - $u_{[go]}(0) = R(0) + 0.1R(0) + 0.9R(1) = 0.9$
  - $u_{[go]}(1) = R(1) + 0.9R(0) + 0.1R(1) = 1.1$
  - This is just the direct reward function (taking into account the probabilistic transitions)



Utility of depth-1 plan given state, outcome of first action, and percept

8 distinct depth-2 plans for each state (16 plans)

Utilities of depth-1 plans

$u_{[stay]}(0) = 0.1 \quad u_{[go]}(0) = 0.9$

$u_{[stay]}(1) = 1.9 \quad u_{[go]}(1) = 1.1$

Reward of state

Probability of next state

Probability of percept

Choose action based on percept ($0 : \boldsymbol{stay}$); receive utility of actual state ($\mathbf{1}$): $u_{[\boldsymbol{stay}]}(\mathbf{1}) = 1.9$

$$u_{[stay,stay/stay]}(0) = \boxed{R(0)} + \Big(0.9 \cdot \underbrace{(\underbrace{0.6 \cdot 0.1}_{\text{percept 0}} + \underbrace{0.4 \cdot 0.1}_{\text{percept 1}})}_{\text{state 0}} + 0.1 \cdot \underbrace{(\underbrace{0.4 \cdot 1.9}_{\text{percept 0}} + \underbrace{0.6 \cdot 1.9}_{\text{percept 1}})}_{\text{state 1}}\Big) = 0.28$$

$$u_{[stay,stay/stay]}(1) = \boxed{R(1)} + \Big(0.1 \cdot (0.6 \cdot 0.1 + 0.4 \cdot 0.1) + 0.9 \cdot (0.4 \cdot 1.9 + 0.6 \cdot 1.9)\Big) = 2.72$$

Sum over states reachable with first action

Sum over possible percepts

$u_{[stay,go/stay]}(0), u_{[stay,stay/go]}(0), u_{[stay,go/go]}(0)$

$u_{[stay,go/stay]}(1), u_{[stay,stay/go]}(1), u_{[stay,go/go]}(1)$

$$u_{[go,stay/stay]}(0) = R(0) + \big(0.1 \cdot (0.6 \cdot 0.1 + 0.4 \cdot 0.1) + 0.9 \cdot (0.4 \cdot 1.9 + 0.6 \cdot 1.9)\big) = 1.72$$

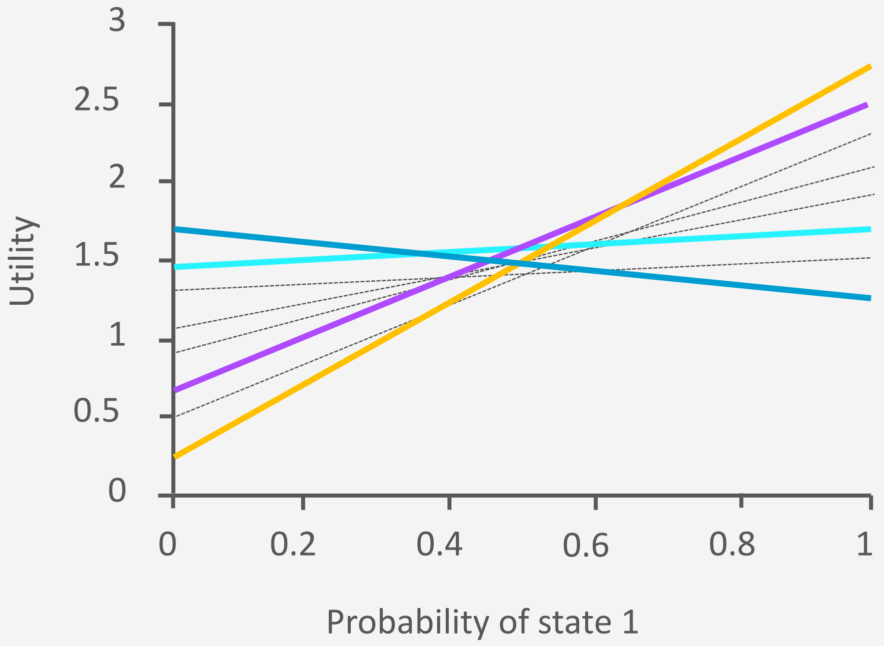
$$u_{[go,stay/stay]}(1) = R(1) + \big(0.9 \cdot (0.6 \cdot 0.1 + 0.4 \cdot 0.1) + 0.1 \cdot (0.4 \cdot 1.9 + 0.6 \cdot 1.9)\big) = 1.28$$

$u_{[go,go/stay]}(0), u_{[go,stay/go]}(0), u_{[go,go/go]}(0)$

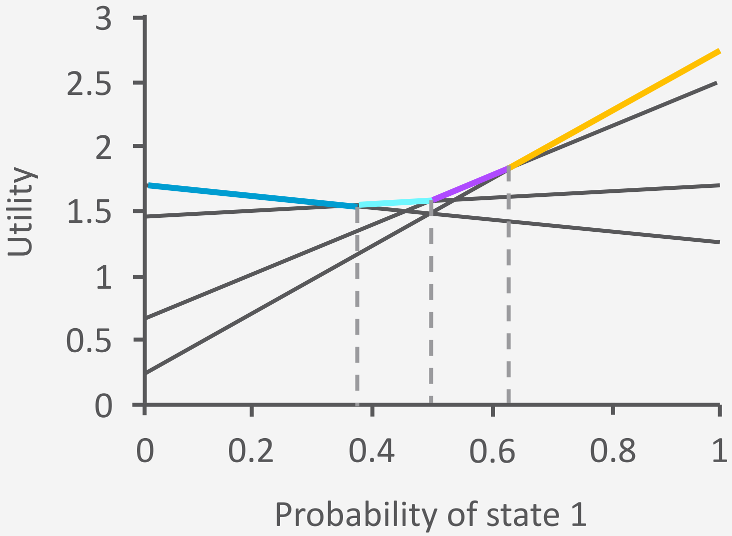
$u_{[go,go/stay]}(1), u_{[go,stay/go]}(1), u_{[go,go/go]}(1)$

Example

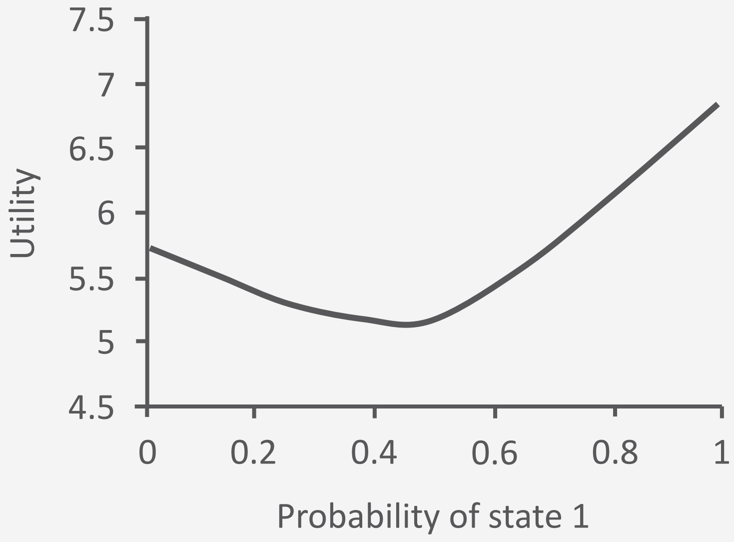
- 8 distinct depth-2 plans for state 1
  - 4 are suboptimal across the entire belief space (dashed lines)
  - With probability $b(1) = 0$
    - $u_{[stay,stay/stay]}(0) = 0.2$
    - $u_{[go,stay/stay]}(0) = 1.7$
  - With probability $b(1) = 1$:
    - $u_{[stay,stay/stay]}(1) = 2.72$
    - $u_{[go,stay/stay]}(1) = 1.28$



Example



Utility of four undominated two-step plans



Utility function for optimal eight step plans

Value Iteration: Algorithm

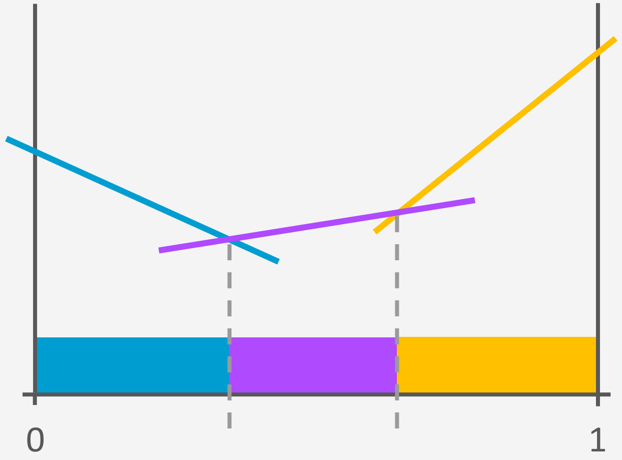
- Returns an optimal set of plans
- Inputs
  - POMDP $pomdp$
    - States $\mathrm{dom}(S)$
    - For all $s \in \mathrm{dom}(S)$,
      - Applicable actions $A(s)$
      - Transition model $P(s'|a,s)$
      - Sensor model $P(o|s)$
      - Rewards $\rho(s)$
    - Discount γ
  - Maximum error allowed ϵ

```
function value-iteration(pomdp,ϵ)
   U′ ← a set containing the empty plan [] with u[](s)=R(s)
   repeat
      U ← U′
      U′ ← the set of all plans consisting of an action and,
             for each possible next percept, a plan in U with
             utility vectors computed as on previous slide
      U′ ← Remove-dominated-plans(U′)
   until Max-difference(U,U′) < ϵ(1-γ)/γ
   return U
```

- Local variables
  - U, U' sets of plans with associated utility vectors u_p

Solutions for POMDP

- Belief MDP has reduced POMDP to MDP
  - MDP obtained has a multidimensional continuous state space
- Extract a policy from utility function returned by value-iteration algorithm
  - Policy $\pi(b)$ can be represented as a set of regions of belief state space
  - Each region associated with a particular optimal action
  - Value function associates distinct linear function of b with each region
  - Each value or policy iteration step refines the boundaries of the regions and may introduce new regions



Intermediate Summary

- POMDP
  - Uncertainty about state → belief state
  - Solving a POMDP = Solving an MDP on space of belief states
  - Policy = conditional plans
  - Value iteration to find optimal policy
    - Very expensive, even with deletion of dominated plans

What to do alternatively? Find sub-optimal plans
- Sampling approaches
- In combination with deep learning methods

Outline: Decision Making – Extensions

Partially Observable Markov Decision Process (POMDP)

- POMDP agent, belief state, belief MDP
- Conditional plans, value iteration

Decentralised POMDP (Dec-POMDP)

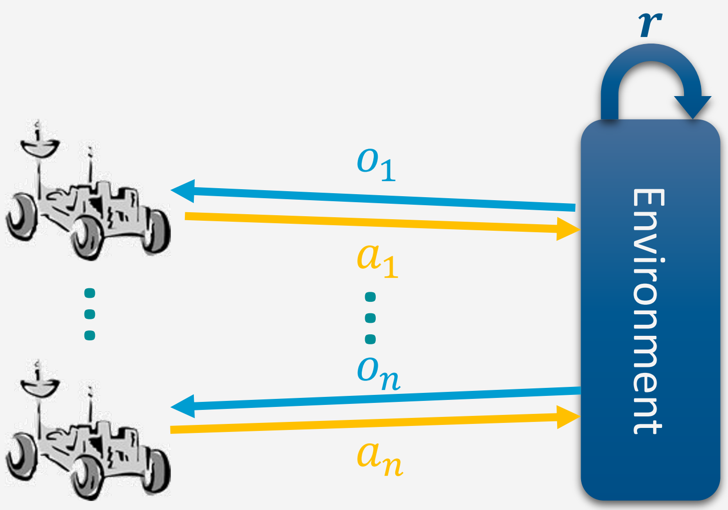
- Dec-POMDP, local policy, joint policy, value function
- Communication, full observability, Dec-MDP
- Solutions for finite, infinite, indefinite horizon

Multi-agent Scenarios

- Ambulance allocation
  - Multiple ambulance services
    - Business oriented operation
    - Competition for government funds and public opinion
  - Given several locations that require medical assistance, how many ambulances from which firm will go to which location?
- Firefighters
  - Maintain effort toward saving the building or draw back and minimise spread of fire?
  - Concentrate on a multitude of smaller fires or allow controlled unification and deal with only one location?
    - Will transportation routes be endangered?
    - Are there still civilians evacuating from the area / building?
  - Push through the fire to victims or save the fire crew and pull out?
    - If multiple crews are on site, which one goes? When?

Setting

- Single and repeated interactions with *joint rewards*: traditional game theory
- Interactions involving *joint state + reward* focus of decision-theory inspired approaches to game theory
  - Extensions of single-agent models to multi-agent settings
- Multi-agent setting
  - Co-operation of agents (team)
    - Vs. self-interested acting (all the way to hostile settings)
  - Problem: planning how to act
    - Joint payoff $\boldsymbol{r}$ but *decentralised* actions a_i and observations o_i
    - Joint state, influenced by actions, can influence rewards
    - Perfect vs. incomplete information about others



Decentralised POMDP (Dec-POMDP)

- Dec-POMDP: tuple $(I, S, \{A_i\}_{i \in I}, \{O_i\}_{i \in I}, T, R, \Omega)$
  - I = a finite set of agents indexed $1, \ldots, N$
  - $\mathrm{dom}(S)$ = a finite set of states
  - A_i = a finite set of actions available to agent $i \in I$
    - $\vec{A} = \bigotimes_{i \in I} A_i$ set of joint actions
  - O_i = a finite set of observations available to agent $i \in I$
    - $\vec{O} = \bigotimes_{i \in I} O_i$ set of joint observations
  - Transition function $T(s', s, \vec{a}) = P(s' | s, \vec{a})$
  - Reward function $R(s)$ or $R(\vec{a}, s)$
  - Sensor model (observation function) $\Omega(\vec{o}, \vec{a}, s) = P(\vec{o} | \vec{a}, s)$
- Co-operative, decision-theoretic setting: Joint reward function R, joint state space S

Generalising Dec-POMDPs

- Partially observable stochastic game (POSG)
  - Dec-POMDP $(I, S, \{A_i\}_{i \in I}, \{O_i\}_{i \in I}, T, \cancel{R}, \Omega)$ but with individual reward functions $\{R_i\}_{i \in I}$
  - Reward function R_i for each agent $i \in I$

- For self-interested or adversarial acting
  - Local optimum not guaranteed to be the global optimum
  - Dominant strategy equilibrium: best response (highest utility) given any state
    - Not guaranteed to exist
    - Prisoner’s dilemma: Dominant strategy not always pareto-optimal strategy
  - Nash Equilibrium: No agent has incentive to change its strategy if no other agent changes its strategy
    - Always exists

Policies for Dec-POMDPs

- Local policy π_i for agent i
  - Representations: Mappings…
    - from local histories of observations $h_i = \left(o_i^{(1)}, \dots, o_i^{(t)}\right)$ over O_i to actions in A_i
    - from local abstraction of joint state s in S to actions in A_i
    - from (generalised) belief states B_i to actions in A_i
      - Belief MDP
    - from internal memory states to actions
- Joint policy $\pi = (\pi_1, \dots, \pi_N)$
  - Tuple of local policies, one for each agent in I

Value Functions for Dec-POMDPs

- Value functions work as before given a joint policy
  - Value of a joint policy π for a finite-horizon Dec-POMDP with initial state $s^{(0)}$

$$V^{\pi}\left(s^{(0)}\right) = E\left[\sum_{t=0}^{h-1} R\left(\vec{a}^{(t)}, s^{(t)}\right) \middle| s^{(0)}, \pi\right]$$

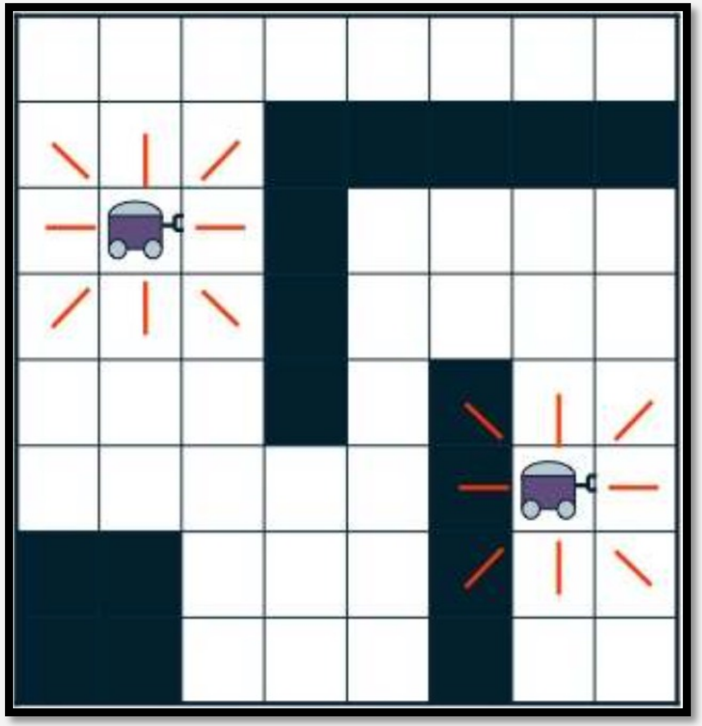
  - Value of a joint policy π for a infinite-horizon Dec-POMDP with initial state $s^{(0)}$ and discount factor $\gamma \in [0,1)$

$$V^{\pi}\left(s^{(0)}\right) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R\left(\vec{a}^{(t)}, s^{(t)}\right) \middle| s^{(0)}, \pi\right]$$

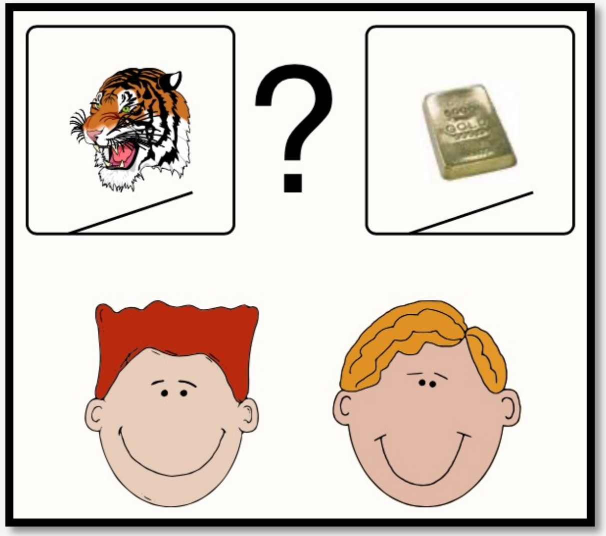
  - $\vec{a}_t$ joint action at time step t

Example: Two-agent Grid World

- Agents: two
- States: grid cell pairs
- Actions: move U, D, L, R, stay
- Transitions: noisy
- Observations: cell occupancy in the directions of the red lines
- Rewards: negative unless sharing the same square



Example: The Dec-Tiger Problem



- A toy problem: *decentralized tiger*
- State space:
  - Position of tiger behind one of two doors (left / right)
  - Treasure behind other door
- Reward:
  - Opening correct door: both receive treasure
  - Opening wrong door: both get attacked by a tiger
- Actions: Agents can open a door, or listen
  - After opening a door, game is reset with tiger behind a randomly chosen door
- Observations: Two noisy observations: hear tiger left or right
- Agents do not know the other’s actions or observations

Worst-case Complexity of DecPOMDP

- Space complexity
  - Transition model: $\mathcal{O}(s \cdot s \cdot a^N)$
  - Sensor model: $\mathcal{O}(s \cdot o^N)$ or $\mathcal{O}(s \cdot o^N \cdot a^N)$
  - Reward function: $\mathcal{O}(s)$ or $\mathcal{O}(s \cdot a^N)$
- Runtime complexity of brute-force search
  - Evaluation cost of a joint policy: $\mathcal{O}\left(s \cdot o^{Nh}\right)$
  - Policy space: $\mathcal{O}\left(a^{\frac{N(o^h-1)}{o-1}}\right)$
- Notations
  - $s = |S|$
    - State space size
  - $a = \max_{i \in I} |A_i|$
    - Largest individual action space size
  - $o = \max_{i \in I} |O_i|$
    - Largest individual observation space size
  - h
    - Horizon
  - N
    - Number of agents

Communication?

- Can make working towards a common goal easier
  - Agents in grid world can communicate their intent (direction of travel)
- Definitely makes the formalism more complicated
  - Dec-POMDP with communication (Dec-POMDP-Com)
    - Dec-POMDP $(I, S, \{A_i\}_{i \in I}, \{O_i\}_{i \in I}, T, R, \Omega)$ defined as before extended with
      - Alphabet Σ for communication
      - $\sigma_i \in \Sigma$ an atomic message sent by agent i
      - $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ a joint message
      - $\varepsilon_\sigma \in \Sigma$ a null message, sent by an agent that does not want to transmit anything to the others (no cost of sending ε_σ)
      - Cost function C_Σ for transmitting atomic message
      - Reward function $R(\vec{a}, s', \vec{\sigma})$ incorporating joint message

New dimensions:
- Do agents always share information?
- Can they intentionally withhold information?
- Can they lie?

Dec-MDP

- Joint full observability
  - Collective observability
  - A DEC-POMDP is jointly fully observable if the N-tuple of observations made by all the agents uniquely determine the current global state
    - That is, if $P(\vec{o}|\vec{a}, s') > 0$, then $P(s'|\vec{o}) = 1$
- Dec-MDP ≙ Dec-POMDP with joint full observability
  - Same as before:
MDP ≙ POMDP with full observability
  - Alternative name: multi-agent MDP

Solving Dec-POMDPs

- Problem: No joint belief available
  - Only partial information about state available to each agent

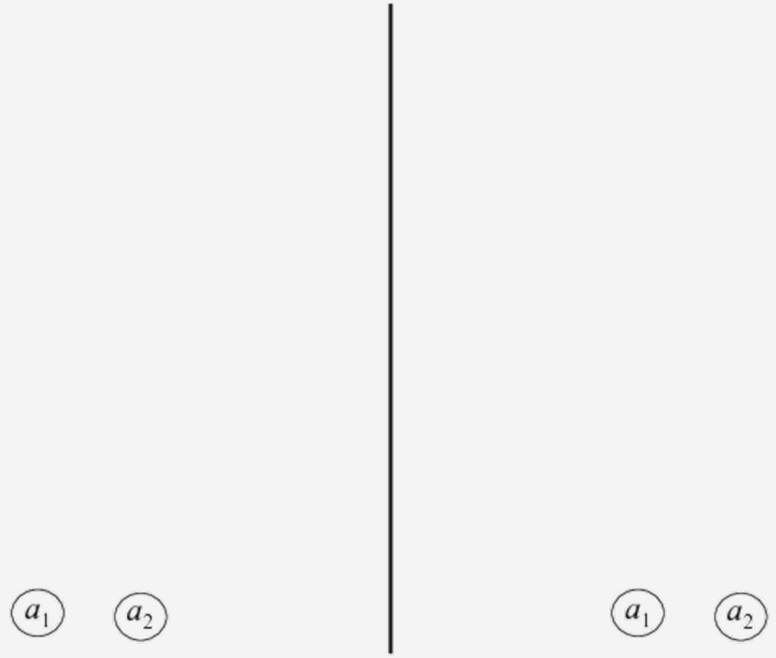
- Complexity: NEXP-complete
  - Optimal solutions using dynamic programming paradigm + exploiting structure if present
  - Reduction to NP when agents mostly independent + communication can be explicitly modelled and analysed
    - Requires that one can factorise the joint state space into a state space for each agent that is mostly independent of all others
    - The same goes for the observations and the reward function

Exhaustive Search

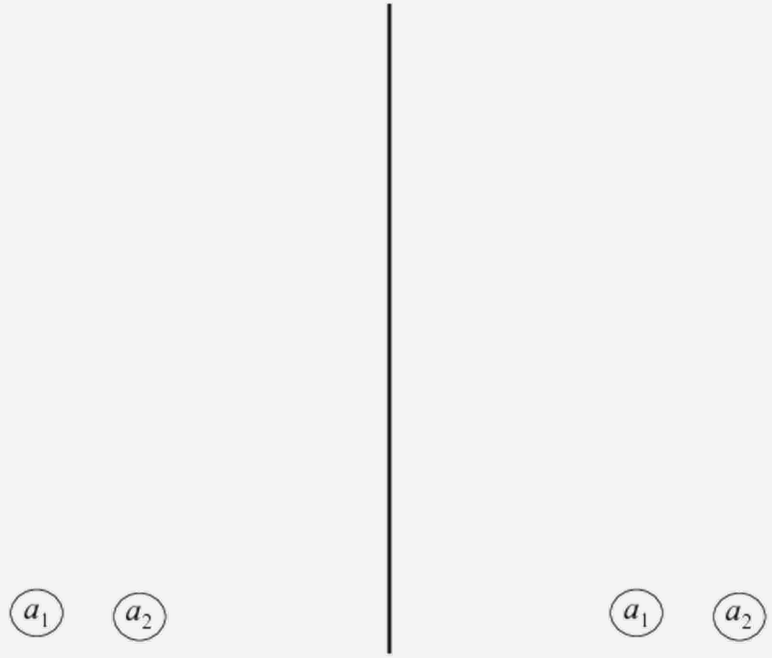
- Optimal solution approach for general models with a finite horizon h
- Procedure:
  - Do a search for each agent to find optimal local policies with a limited depth of h
  - Prune dominated search paths/strategies locally by considering the joint state and other agents' policies (globally)
    - Requires central oversight
    - Cannot be done locally without a huge amount of communication
- Even with pruning, still limited to small problems

Exhaustive Search and Pruning

Without Pruning

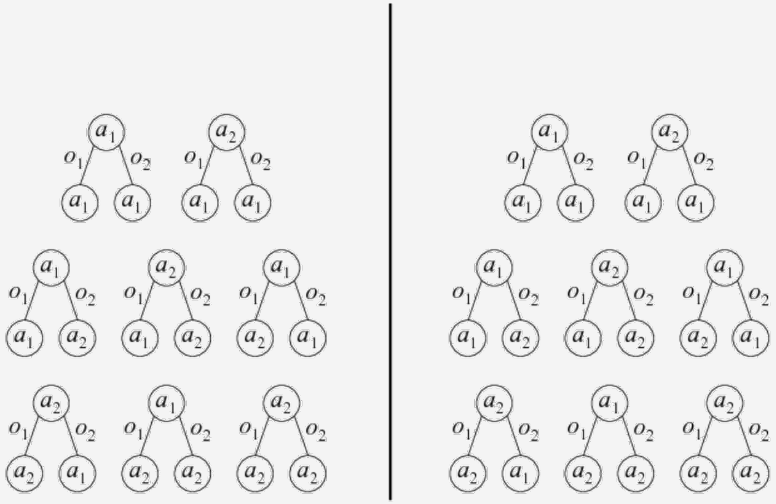


With Pruning

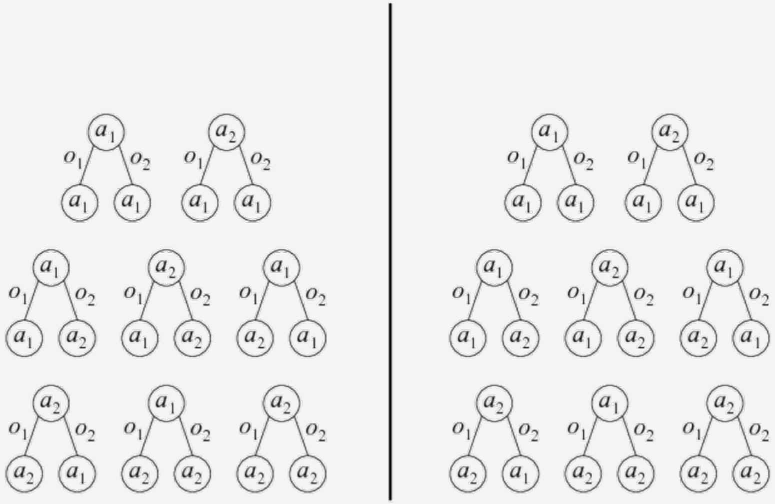


Exhaustive Search and Pruning

Without Pruning

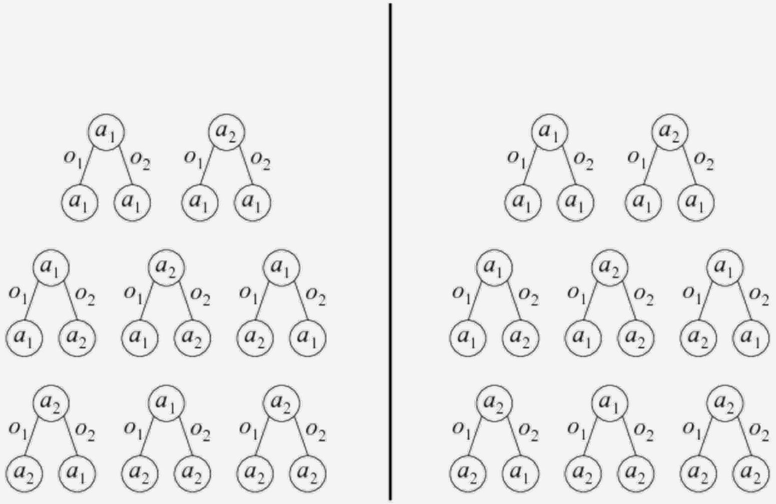


With Pruning

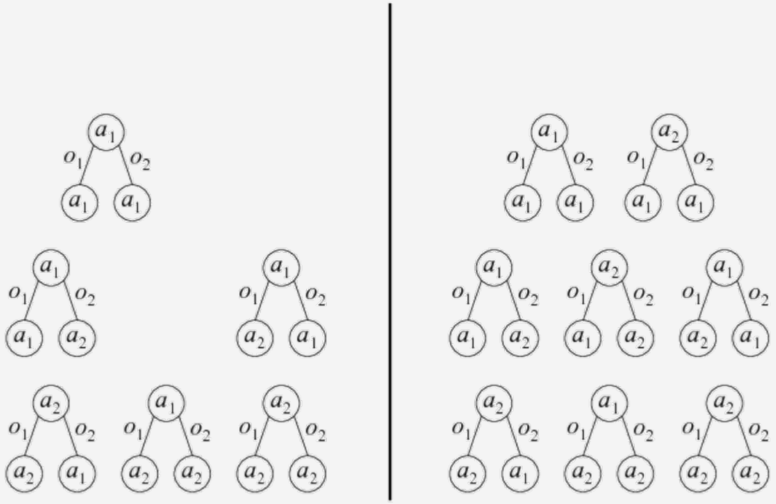


Exhaustive Search and Pruning

Without Pruning

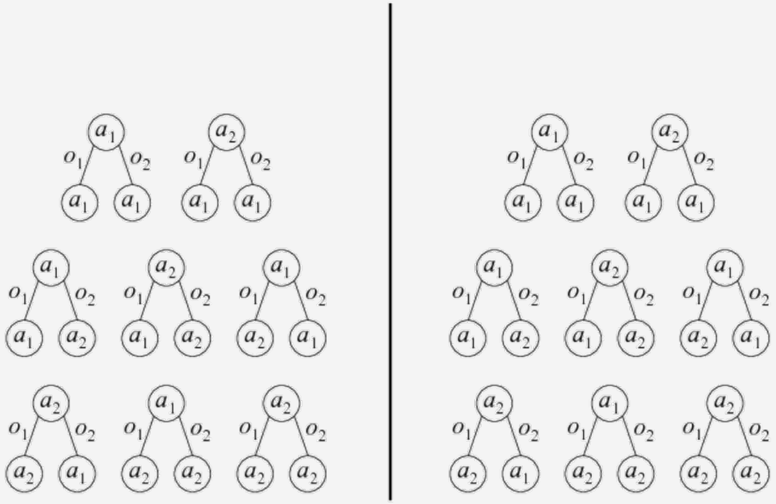


With Pruning

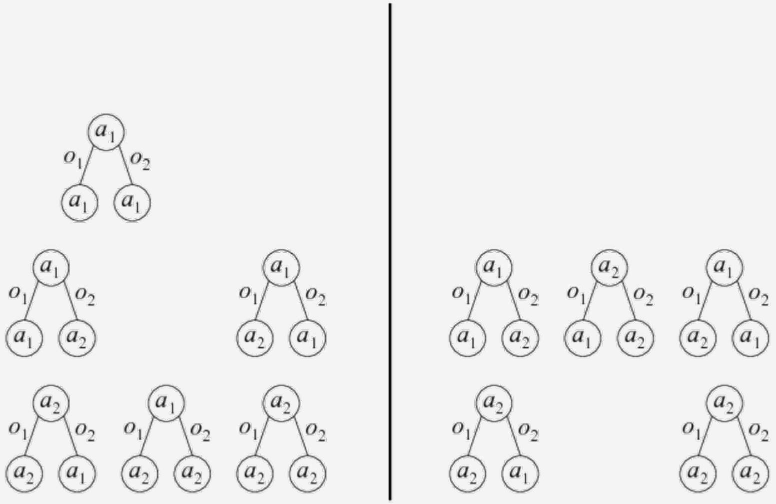


Exhaustive Search and Pruning

Without Pruning

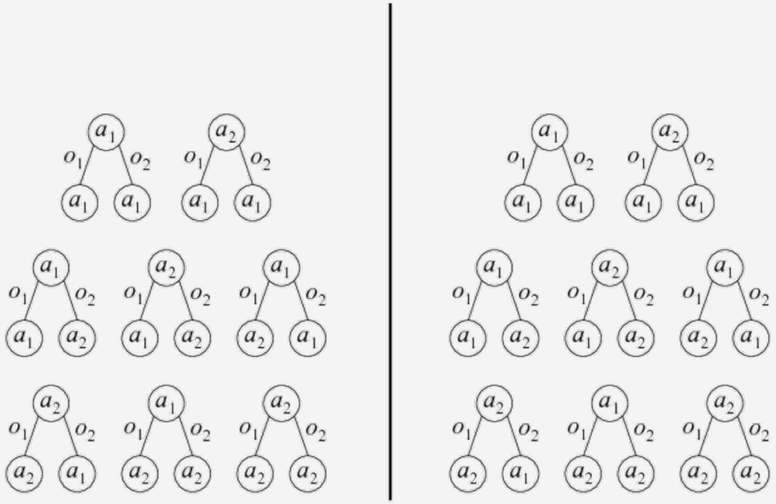


With Pruning

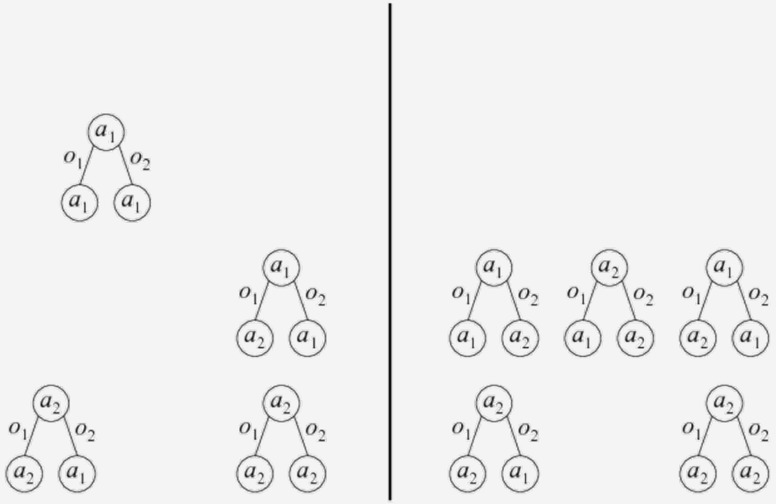


Exhaustive Search and Pruning

Without Pruning

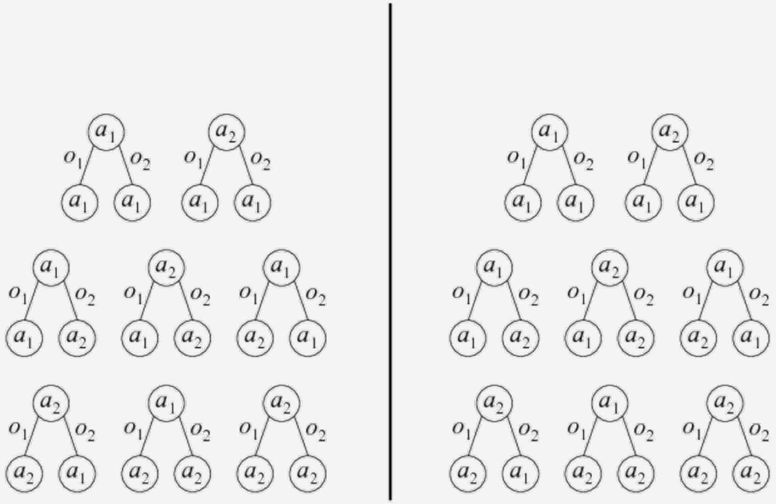


With Pruning

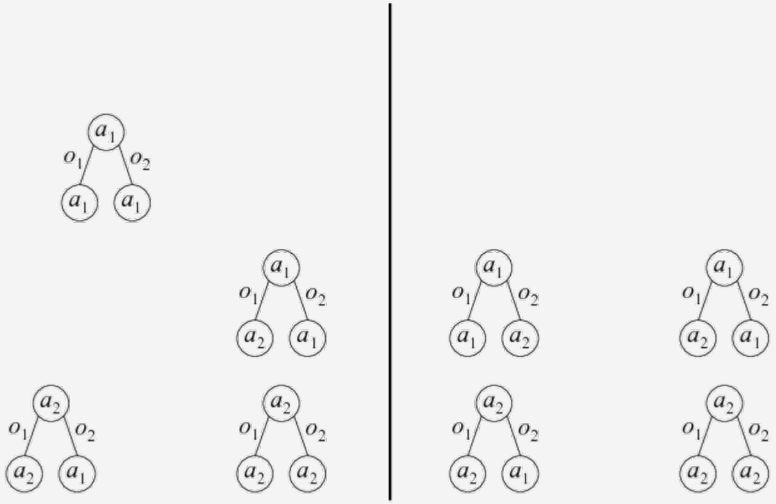


Exhaustive Search and Pruning

Without Pruning

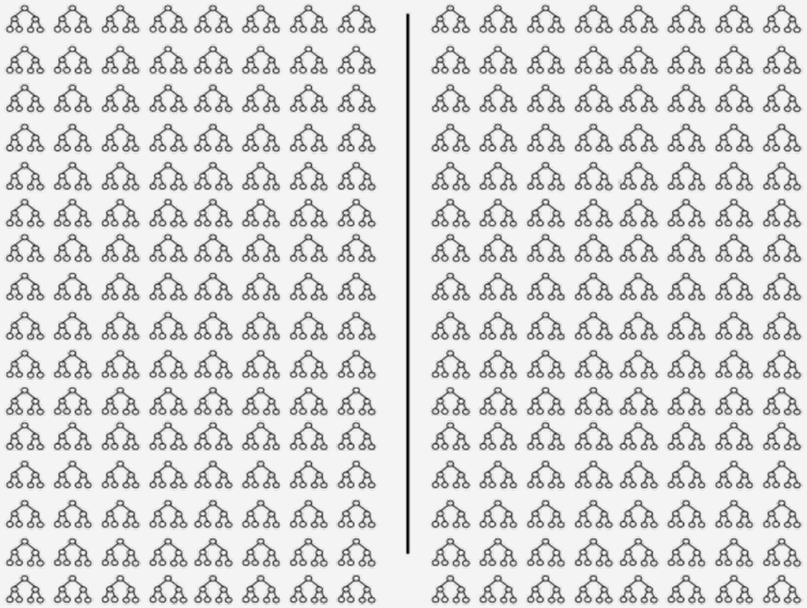


With Pruning

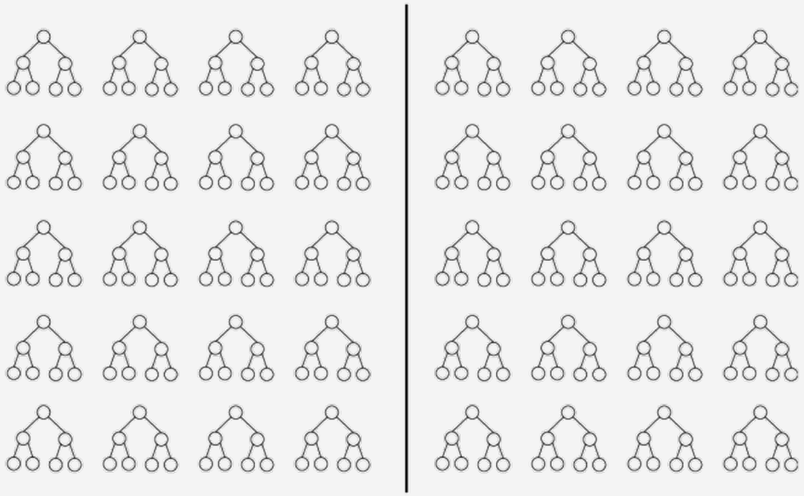


Exhaustive Search and Pruning

Without Pruning



With Pruning



Joint Equilibrium Search for Policies

- Approximate solution approach for general models with a finite horizon h
  - Input:
    - DecPOMDP $(I, S, \{A_i\}_{i\in I}, \{O_i\}_{i\in I}, T, R, \Omega)$
    - Horizon h
    - Possibly error margin ε
- Instead of exhaustive search, find best response
  - Nash equilibrium: no agent has incentive to change its policy if no other agent changes its policy
  - Convergence criterion needed
    - E.g., no change (or only ε change) in any policy
  - Same worst-case complexity, but in practice much faster
    - Can include pruning, further heuristics when looking for best response policy

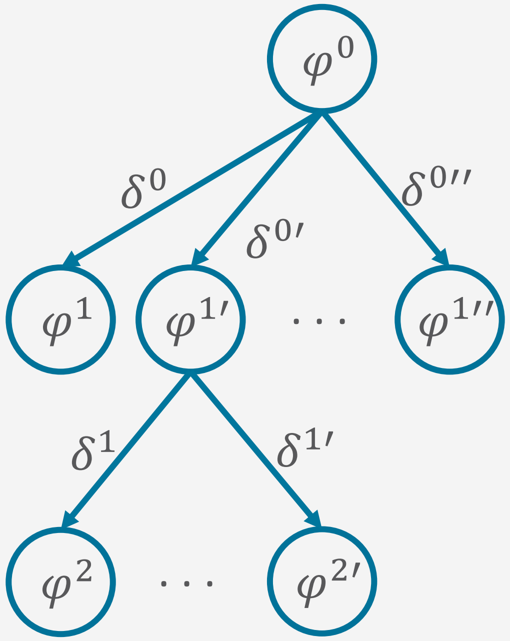
```
JESP(dec-pomdp, h)
    while not converged do
        for i = 1 to n do
            Fix other agent policies
            Find a best response policy for agent i
```

Turns DecPOMDP into a POMDP for i

Multi-agent A* (MAA*)

- Optimal solution approach for general models with a finite horizon h
  - Inputs:
    - DecPOMDP $(I, S, \{A_i\}_{i \in I}, \{O_i\}_{i \in I}, P_{tr}, R, P_{obs})$
    - Horizon h
    - Heuristics $\hat{V}(\varphi^t)$
- A*-like search over partially specified joint policies
  - $\varphi^t = (\delta^0, \dots, \delta^{t-1})$
  - $\delta^t = (\delta_0^t, \dots, \delta_n^t)$
  - $\delta_i^t : \vec{O}_i^t \rightarrow A_i$
- Requires an admissible heuristic function $\hat{V}(\varphi^t)$

$$\underbrace{\hat{V}(\varphi^t)}_{F} = \underbrace{V^{0\dots t-1}(\varphi^t)}_{G} + \underbrace{\hat{V}^{t\dots h-1}(\varphi^t)}_{H}$$



How to Get a Heuristic Function?

- Solve simplified settings, e.g.,
  - Solve the underlying MDP (approximately or optimally) given assumptions:
    - Centralised observations
    - Full observability
      - Simulate / sample unobserved values
  - Solve a belief MDP given assumption
    - Centralised observations
- Domain-specific heuristics

Memory Bounded Search

- Approximate solution approach for general models with a finite horizon h
  - Inputs:
    - DecPOMDP $(I, S, \{A_i\}_{i \in I}, \{O_i\}_{i \in I}, T, R, \Omega)$
    - Horizon h
- Do not keep all policies at each step but a fixed number for each agent $maxTrees$
  - Select $maxTrees$ in a way that $maxTrees \cdot |I|$ trees fit into memory
    - Can be difficult to choose; often small in practice
  - Select trees by using heuristic (like A*)

```
MBDP(dec-pomdp, h)
    Start with a one-step policy for each agent
    for t = h downto 1 do
        Backup each agent's policy
        for k = 1 to maxTrees do
            Compute heuristic policy and resulting
                    belief state b
            Choose best set of trees starting at b
    Select best set of trees for initial state b0
```

MBDP =
Memory
Bounded
Dynamic
Programming

Infinite Horizon

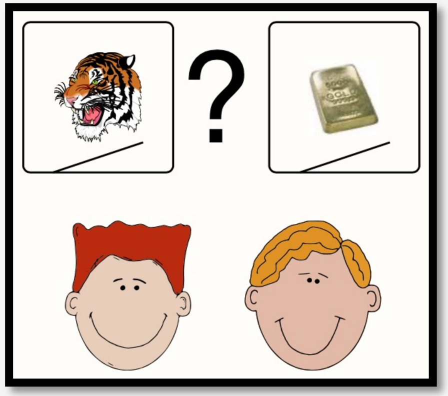
- Approximate using a large enough horizon h
  - Neither efficient, nor compact
- Selection of solution approaches based on solution approaches already seen for MDPs / POMDPs:
  - Policy iteration
    - Start with one-step plans, extend further
    - Automata-based approaches (Moore/Mealy automata to represent policy)
    - Intractable for all but the smallest problems
  - Best-first search
    - Finds optimal fixed-size solutions; use start state info
    - High search time $\rightarrow$ small sizes only
- Further solution approaches use non-linear programming

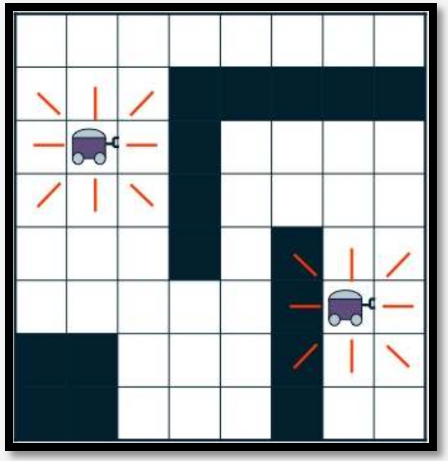
Indefinite Horizon

- Many natural problems terminate after a goal is reached
  - Meeting or catching a target
  - Cooperatively completing a task
- Unclear how many steps are needed until termination
- Under certain assumptions can produce an optimal solution
  - E.g., terminal actions and negative rewards
    - Such as the 4x3 grid:
      terminal states, negative rewards for all but one terminal state
- Otherwise, can bound the solution quality by sampling

Benchmark Problems

- *DEC-Tiger*
  - (Nair et al., 2003)
- BroadcastChannel
  - (Hansen et al., 2004)
- *Meeting on a grid*
  - (Bernstein et al., 2005)
- Cooperative Box Pushing
  - (Seuken and Zilberstein, 2007a)
- Recycling Robots
  - (Amato et al., 2007)
- FireFighting
  - (Oliehoek et al., 2008b)
- Sensor network problems
  - (Nair et al., 2005; Kumar and Zilberstein, 2009a,b)





Software for Dec-POMDPs

- The *MADP toolbox* aims to provide a software platform for research in decision-theoretic multiagent planning
  (Spaan and Oliehoek, 2008)
- Main features:
  - Uniform representation for several popular multiagent models
  - Parser for a file format for discrete Dec-POMDPs
  - Shared functionality for planning algorithms
  - Implementation of several Dec-POMDP planners
- Released as free software, with special attention to the extensibility of the toolbox
- Provides benchmark problems
  - Such as on the previous slide

```
agents: 2
discount: 1
values: reward
states: tiger-left tiger-right
start:
uniform
actions:
listen open-left open-right
listen open-left open-right
observations:
hear-left hear-right
hear-left hear-right
```

```
# Transitions
T: * :
uniform
T: listen listen :
identity
# Observations
O: * :
uniform
O: listen listen : tiger-left : hear-left hear-left : 0.722
O: listen listen : tiger-left : hear-left hear-right : 0.1275
[...]
O: listen listen : tiger-right : hear-left hear-left : 0.0225
# Rewards
R: listen listen : * : * : * : -2
R: open-left open-left : tiger-left : * : * : -50
[...]
R: open-left listen: tiger-right : * : * : 9
```

```
#include "ProblemDecTiger.h"
#include "JESPExhaustivePlanner.h"
int main()
{
   ProblemDecTiger dectiger;
   JESPExhaustivePlanner jesp(3,&dectiger);
   jesp.Plan();
   std::cout
        << jesp.GetExpectedReward()
                << std::endl;
   std::cout
        << jesp.GetJointPolicy()->SoftPrint()
                << std::endl;
   return(0);
}
```

Dec-Tiger Problem Specification and Program

Interim Summary

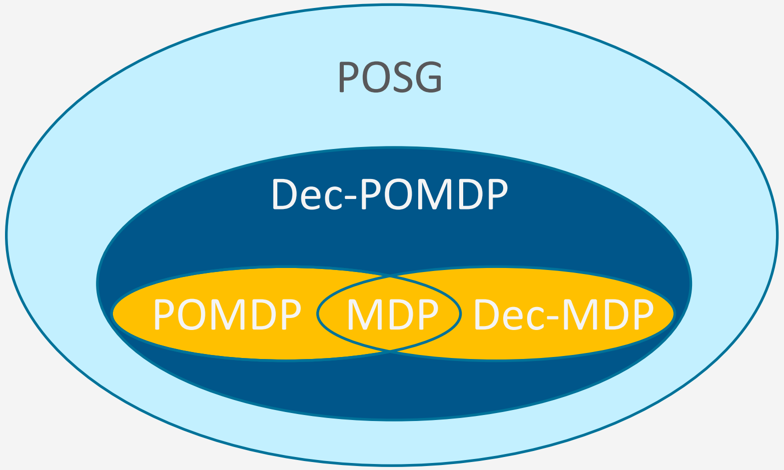
- Dec-POMDPs
  - Local policies, joint policy, value functions
  - Communication, full observability, Dec-MDP
- Solutions for
  - Finite horizon
  - Infinite horizon
  - Indefinite horizon
- MADP toolbox
  - Benchmark problems

Hierarchy of Formalisms

- Most general: *POSG*
  - Set of agents, individual reward functions
  - Environment only partially observable
- Specifications
  1. Decentralisation
     - Joint reward function

  2a. Observable environment

  2b. Multi to single agent
- Most specific: *MDP*
  - One agent, (therefore) one reward function
  - Observable environment



Modelling without any structure in the agent set or state space → potential for efficiency wasted

Outline: Decision Making – Extensions

Partially Observable Markov Decision Process (POMDP)

- POMDP agent, belief state, belief MDP
- Conditional plans, value iteration

Decentralised POMDP (Dec-POMDP)

- Dec-POMDP, local policy, joint policy, value function
- Communication, full observability, Dec-MDP
- Solutions for finite, infinite, indefinite horizon

$\Longrightarrow$ Next: Decision Making – Structure