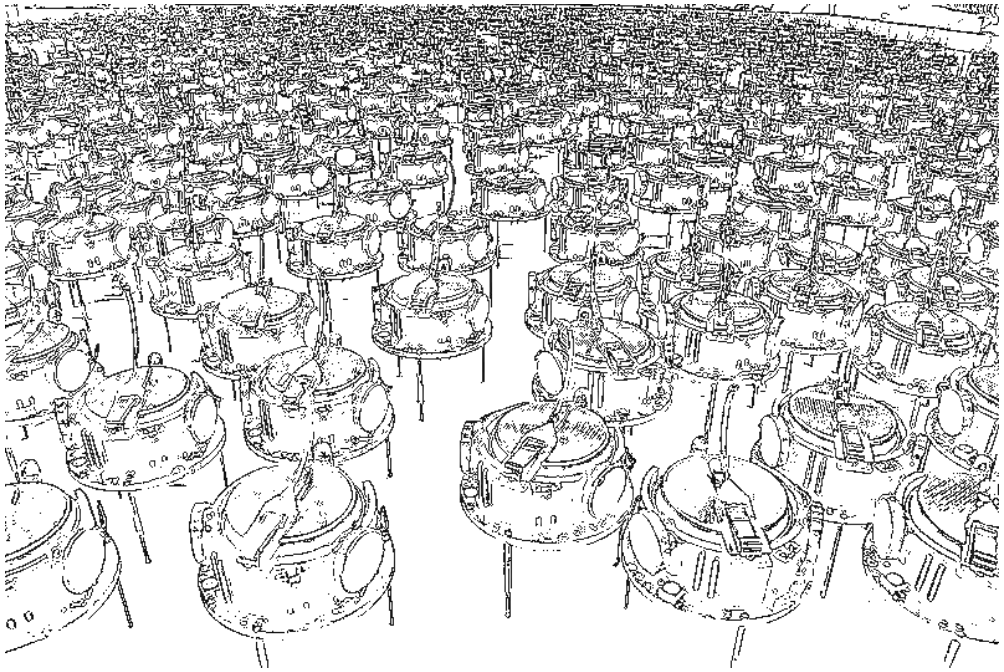
Intelligent Multi Agent Systems



Decentralized POMDPs

Gerhard Neumann

Agenda

So far, we know how to plan and learn with partial observations and a single agent.

- What about multiple agents?
- Each agent gets a local set of observations
- We will still assume that all agents share the same reward
- Decision making is decentralized

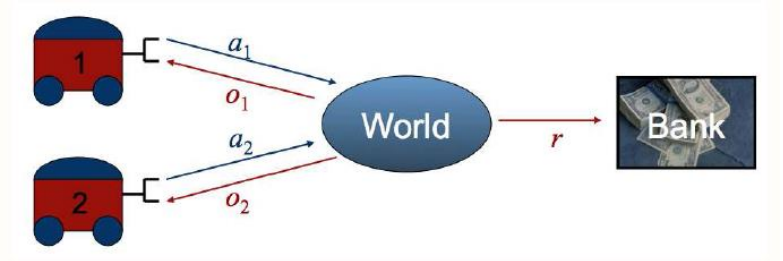
Such a model is called a Decentralized POMDP (Dec-POMDP)!

- There is not much learning out there…
- We will only consider planning

Outline

- Problem Formulation
- Finite Horizon approaches:
  - Bottom-Up
  - Top-Down
- Infinite Horizon approaches
  - Optimizing finite state controllers

Dec-POMDPs



A Dec-POMDP is defined by:

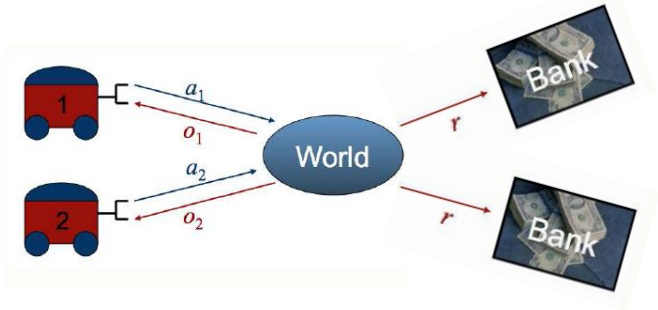
- its state space $\boldsymbol{s} \in \mathcal{S}$
- An action space $\mathcal{A}_i$ for agent *i*
- An observation space O_i for agent *i*
- its transition dynamics $\mathcal{P}(s'|s,a) = \mathcal{P}^a_{s's}$
- observation model per agent $\mathcal{O}_i(\boldsymbol{o}|\boldsymbol{s}) = \mathcal{O}^i_{os}$
- A shared reward function for all agents $r(\boldsymbol{s},\boldsymbol{a}) = \mathcal{R}^a_s$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$

There is a common goal (reward)

We do not know what the other agents observed

Partially Observable Stochastic Games

A Partially Observable Stochastic Game (POSG) is defined by:

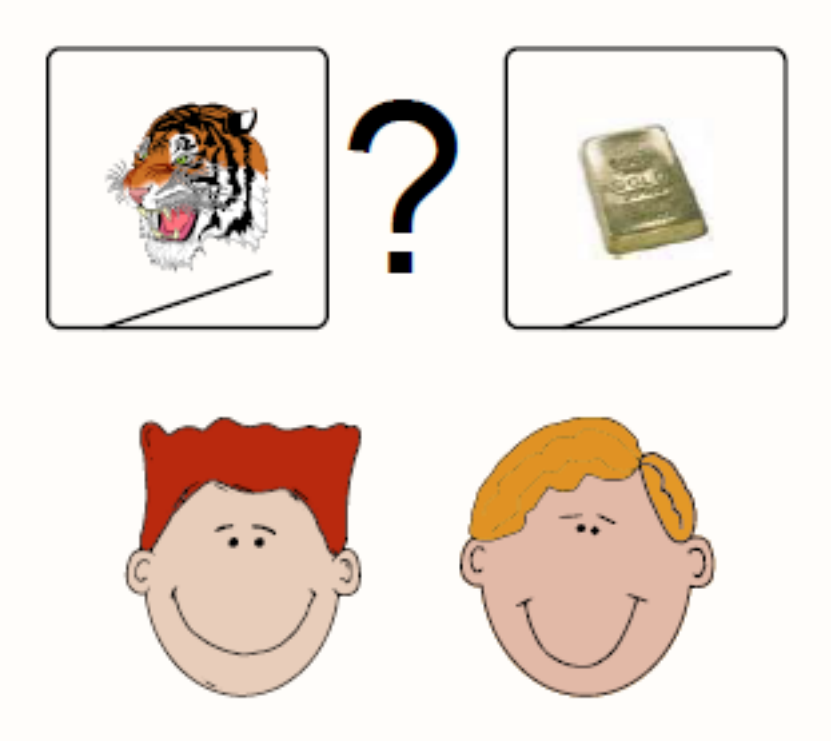


- its state space $\boldsymbol{s} \in \mathcal{S}$
- An action space $\mathcal{A}_i$ for agent *i*
- An observation space O_i for agent *i*
- its transition dynamics $\mathcal{P}(s'|s,a) = \mathcal{P}^a_{s's}$
- observation model per agent $\mathcal{O}_i(\boldsymbol{o}|\boldsymbol{s}) = \mathcal{O}^i_{os}$
- reward function per agent $r_i(\boldsymbol{s},\boldsymbol{a}) = \mathcal{R}^i_{sa}$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$

Competitive agents -> That's the hardest case

Dec-Tiger Problem

- A toy problem: decentralized tiger (Nair et al., 2003).
- Opening correct door: both receive treasure.
  - More treasue if both open at the same time
- Opening wrong door: both get attacked by a tiger.
  - Less painful if both open at the same time
- Agents can open a door, or listen.
- Two noisy observations:
  - hear tiger left or right.
  - Don’t know the other’s actions or observations.



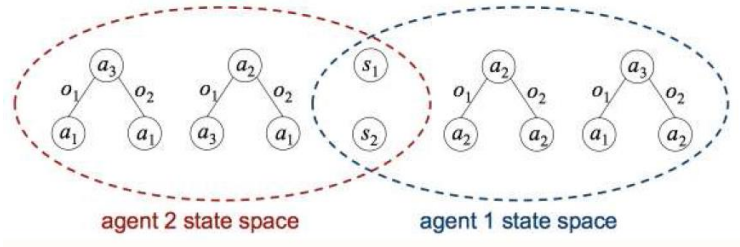
Dec-POMDP policies

Definition: A **local policy** π_i for agent *i*, is a mapping from local histories of observations $\boldsymbol{o}^i = (o_1^i, \ldots, o_t^i)$ over Ω_i^* to actions in A_i , $\pi_i : \Omega_i^* \rightarrow A_i$

Definition: A **joint policy**, $\langle \pi_1, \ldots, \pi_n \rangle$ is a tuple of local policies, one for each agent

Other forms of policy representations:

- Mapping from (generalized) belief states to actions:



- Mapping from internal memory states to actions (finite state controller

DEC-POMDPs complexity

How hard is this problem in comparision to other models?

Some complexity results:

Finite Horizon

MDP	P-complete (if $h < \|S\|$)	(Papadimitriou and Tsitsiklis, 1987)
POMDP	PSPACE-complete (if $h < \|S\|$)	(Papadimitriou and Tsitsiklis, 1987)

Infinite Horizon Discounted

MDP	P-complete	(Papadimitriou and Tsitsiklis, 1987)
POMDP	undecidable	(Madani et al., 1999)

PSPACE-complete if it can be solved using an amount of memory that is polynomial in the input length

DEC-POMDPs complexity

Intuition

- Agents must consider the choices of all others in addition to the state and action uncertainty present in POMDPs.
- In order to know the future choices of other agents we have to reason about the observations of the other agents
- Observation histories of the other agent is our unobserved state

Theorem 2. (Bernstein et al., 2002) Two-agent finite-horizon DEC-POMDPs are NEXP-hard.

- Double exponential!
- Thus provably intractable (unlike POMDP)
- Probably doubly exponential (unlike POMDP)

NEXP problems can be solved by a non-deterministic Turing machine using time $2^{n^{O(1)}}$ and unlimited space.

Outline

Problem Formulation

Finite Horizon approaches

- Bottom-Up
- Top-Down

Infinite Horizon approaches

- Optimizing finite state controllers

Optimal DEC-POMDP solutions

Reminder:

- No beliefs over states available.
- No piecewise linear convex value functions.

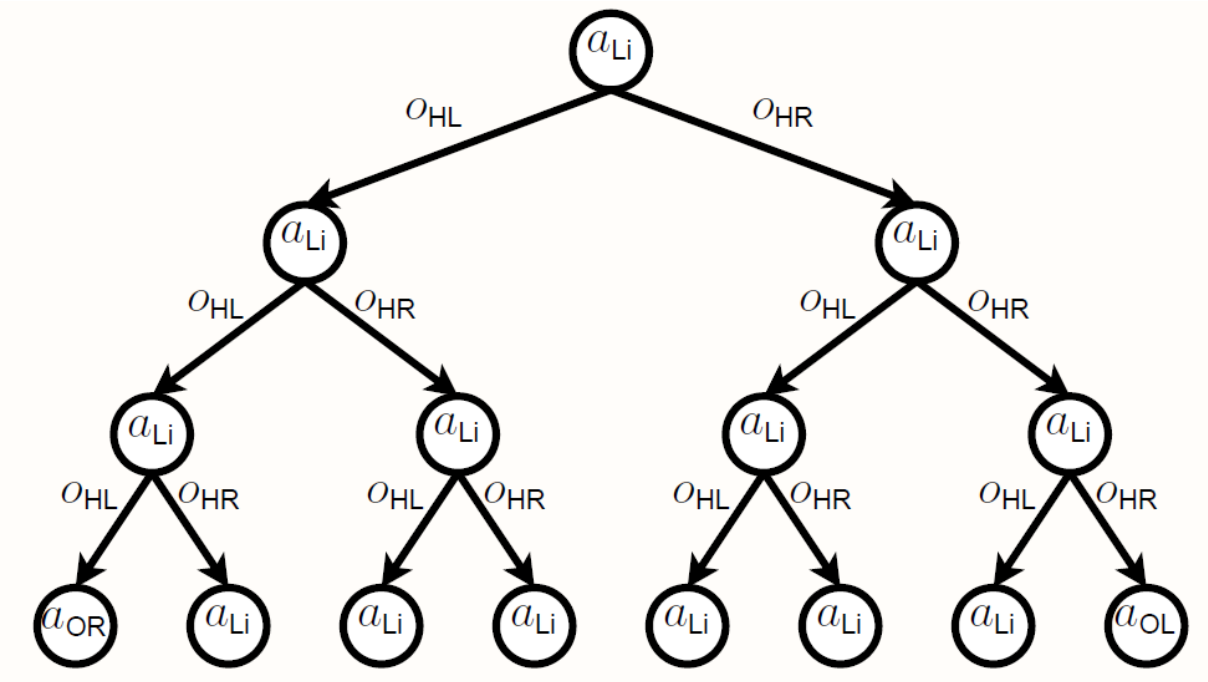
MDP/POMDP algorithms do not transfer directly. . .

- . . . but ideas do transfer

Optimal policies

Naive approach: Evaluate the expected value all policies of horizon T, take the best one

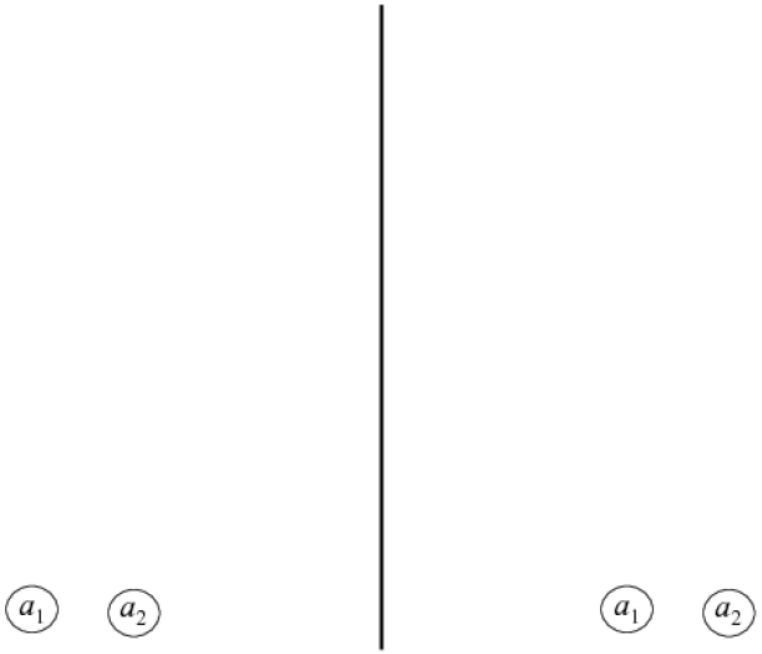
Example: Optimal Policy, Dec-Tiger, T = 4



Exhaustive Search via Dynamic Programing

- Construct all possible policies for the set of agents
- Do this in a bottom up fashion
- Trivially includes an optimal set of trees when finished
- Check all possible combination of trees

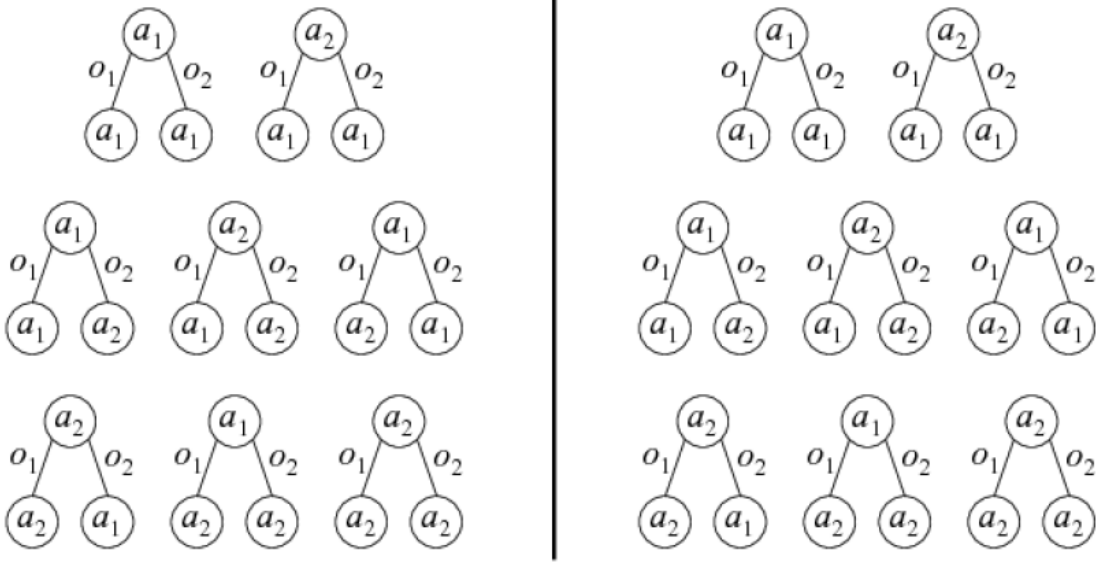
t = 1



Exhaustive Search

- Construct all possible policies for the set of agents
- Do this in a bottom up fashion
- Trivially includes an optimal set of trees when finished
- Check all possible combination of trees

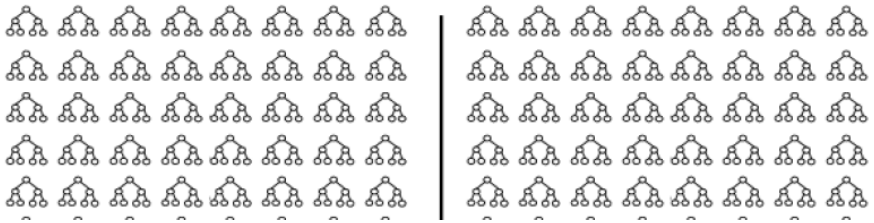
t = 2



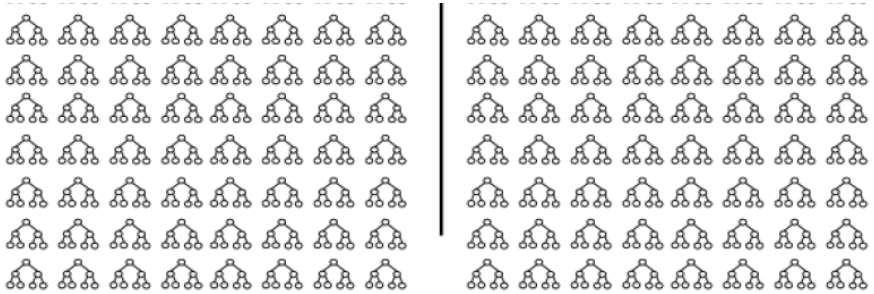
Exhaustive Search

- Construct all possible policies for the set of agents
- Do this in a bottom up fashion
- Trivially includes an optimal set of trees when finished
- Check all possible combination of trees

t = 3



- Number of nodes grows quadratically (2 observations) with the depth
- Number of trees depends exponentially on the number of nodes!



Exhaustive Search

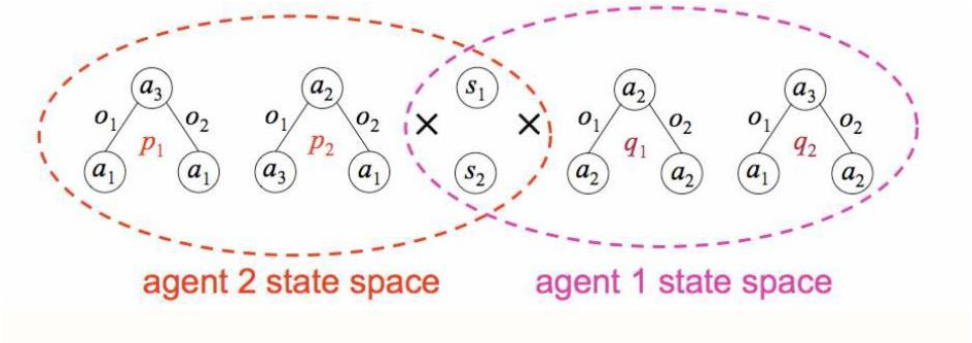
- Can find an optimal set of trees
- Number of each agent's trees grows exponentially at each step
- Many trees will not contribute to an optimal solution
- Can we reduce the number of trees we consider?

Add a pruning step to exhaustive search (Hansen et al., 2004)

- At each step, remove trees that will never be part of optimal solution
- Uses linear programming (LP) to prune these dominated trees
- Ensures any policy that can contribute to optimal is not removed

Pruning Trees

- Prune over multi-agent belief space (policies of the other agents and states of the system)
- Retains optimal policy for any belief state



Linear Programs for Prunning Trees

Variables: $\epsilon, x(\hat{q}_i)$

Objective: Maximize ϵ

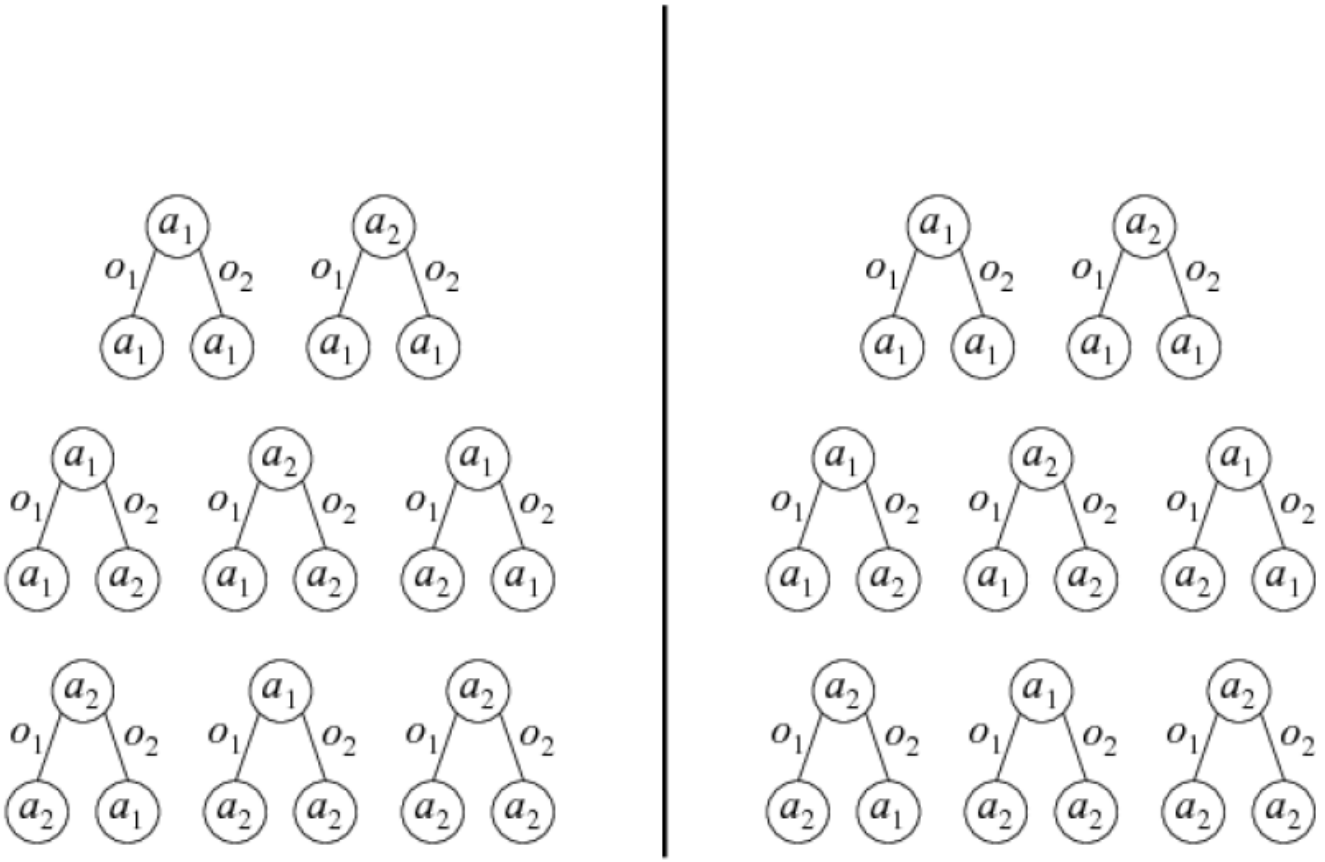
Improvement constraints:

$$\forall s, q_{-i} \quad V(s, q_i, q_{-i}) + \epsilon \leq \sum_{\hat{q}_i} x(\hat{q}_i) V(s, \hat{q}_i, q_{-i})$$

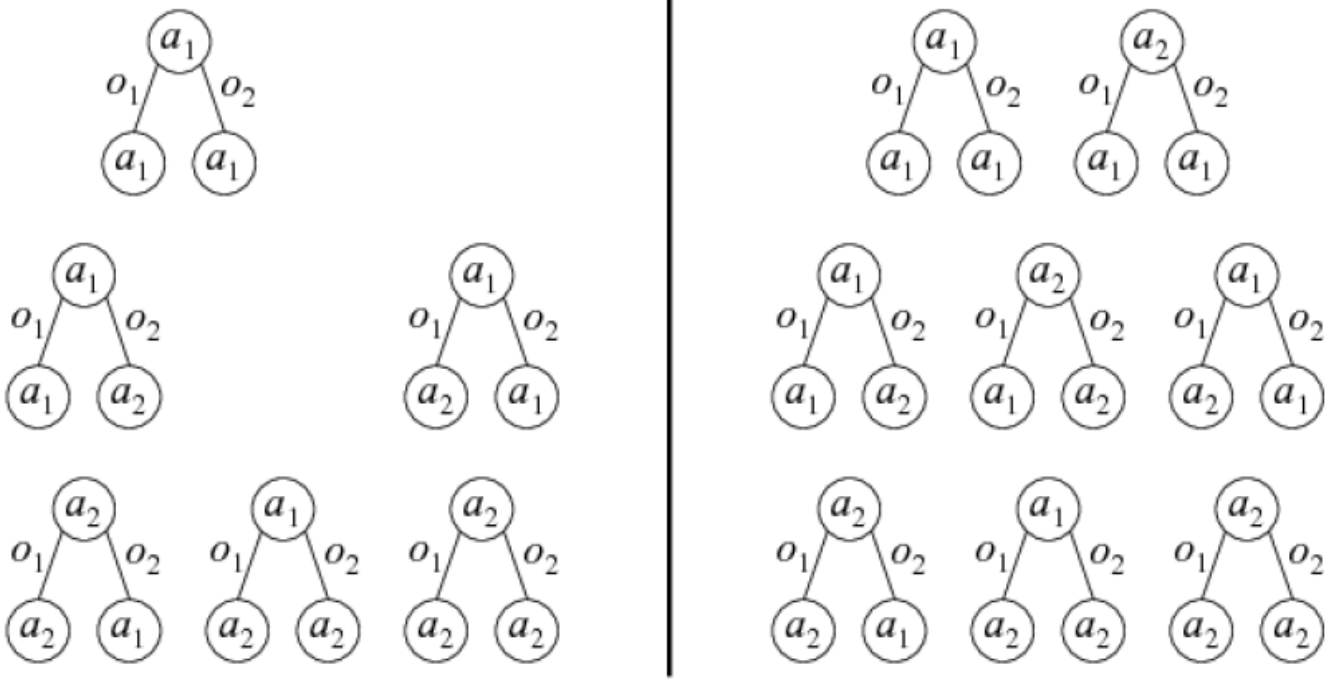
Probability constraints: $\sum_{\hat{q}_i} x(\hat{q}_i) = 1, \quad \forall \hat{q}_i \quad x(\hat{q}_i) \geq 0$

- Prune tree q_i if there is distribution $x(\hat{q}_i)$ of other trees that has a higher value for all system states and trees of the other agents
- This is the case if $\epsilon > 0$

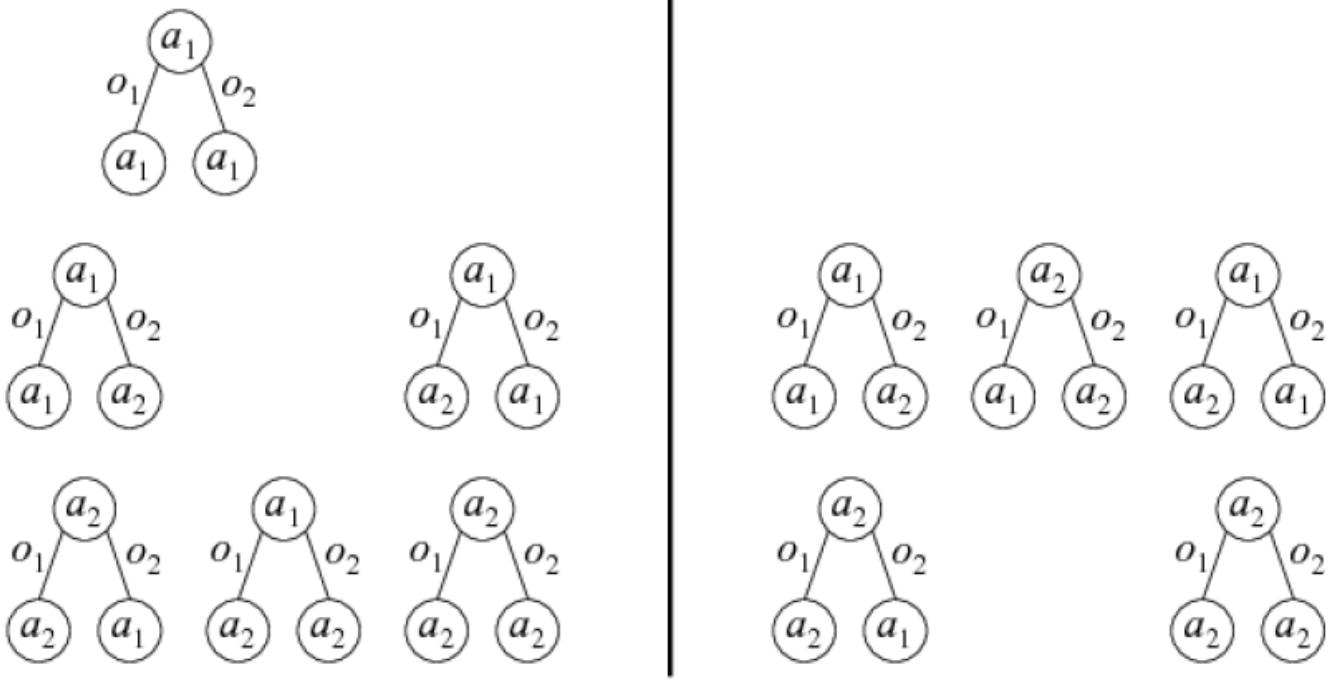
Pruning Trees



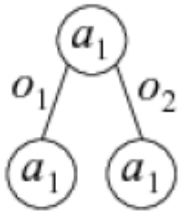
Pruning Trees

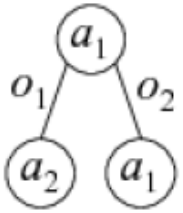


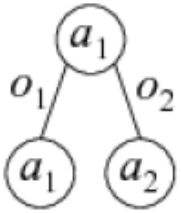
Pruning Trees

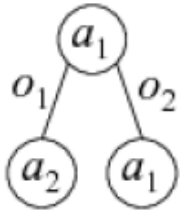


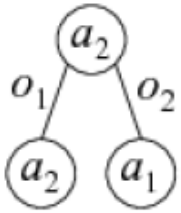
Pruning Trees

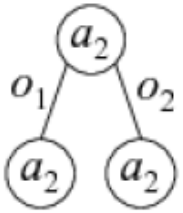


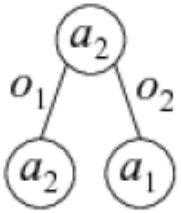


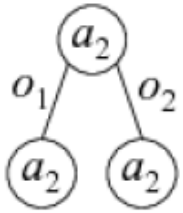




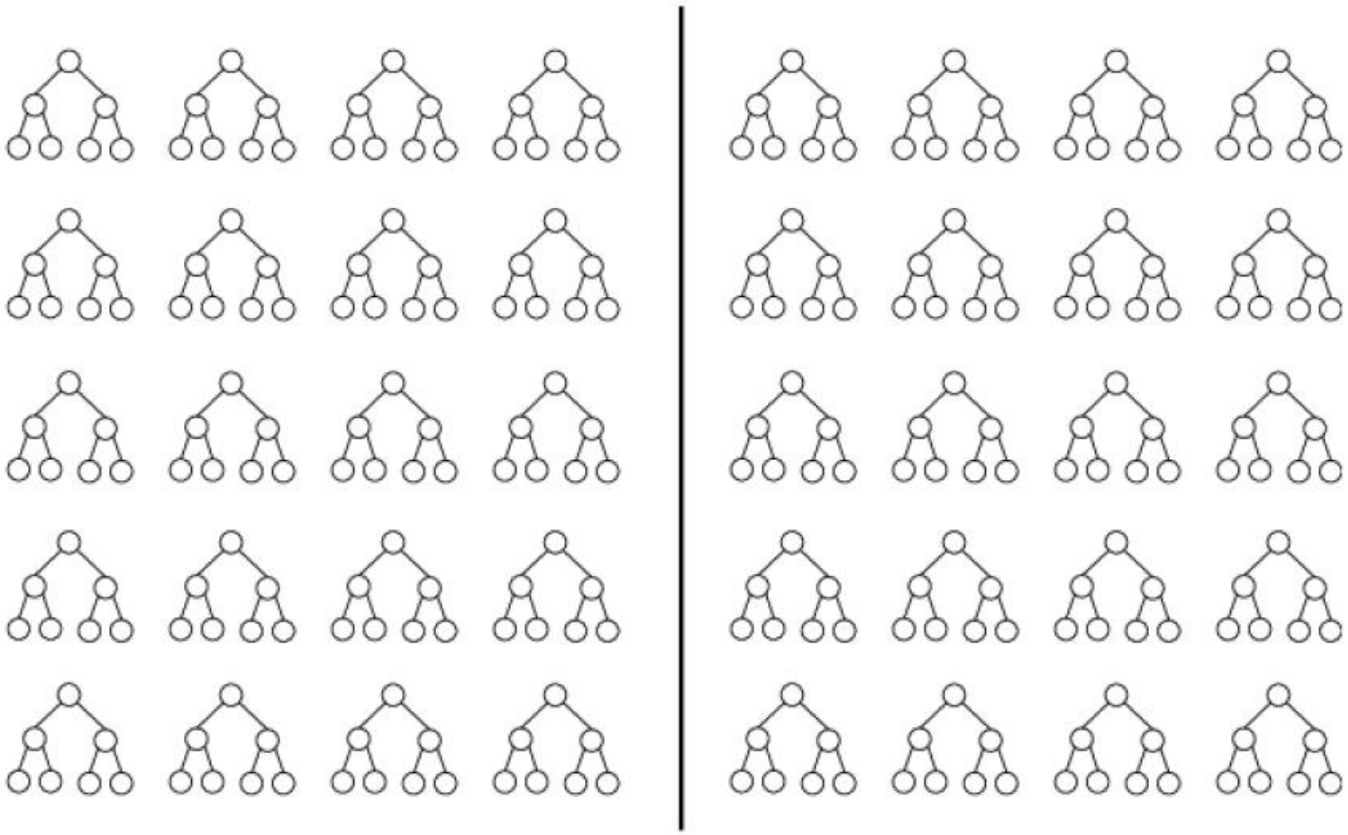








Pruning Trees



Dynamic programming summary

- Will produce optimal solution for any initial state
- Improves scalability of exhaustive backups with pruning at each step
- Still limited to small problems
- For POSGs, iterative removal of very weakly dominated strategies

Joint Equilibrium Search for Policies

Instead of using exhaustive search,

- always keep policies of all but one agent fixed
- and use best-response to update the policy of the particular agent

Algorithm : JESP (Nair et al., 2003)

Start with policy for each agent
while *not converged* **do**
  for $i = 1$ *to* n **do**
    Fix other agent policies
    Find a best response policy for agent i

- Only finds local optimum

Joint Equilibrium Search for Policies

Finding the best-response can be reduced to a complex POMDP problem

- The agent needs to reason about the hidden state AND about the observations of the other agents!
- Define new hidden state $\hat{s}_i = (s, h_{-i})$
  - h_{-i} histories of all other agents

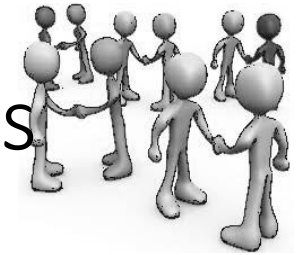
Define new POMDP:

- Transition function:

$$\mathcal{P}(\hat{s}_{i,t+1}|\hat{s}_{i,t}, a_{i,t}) = p(s_{t+1}|s_t, (\pi_{-i}(h_{-i,t}), a_{i,t})) \prod_{j \neq i} O_j(o_{j,t+1}|s_{t+1})$$

- Observation model: equivalent to single agent observation model in Dec-POMDP

Joint Equilibrium Search for Policies



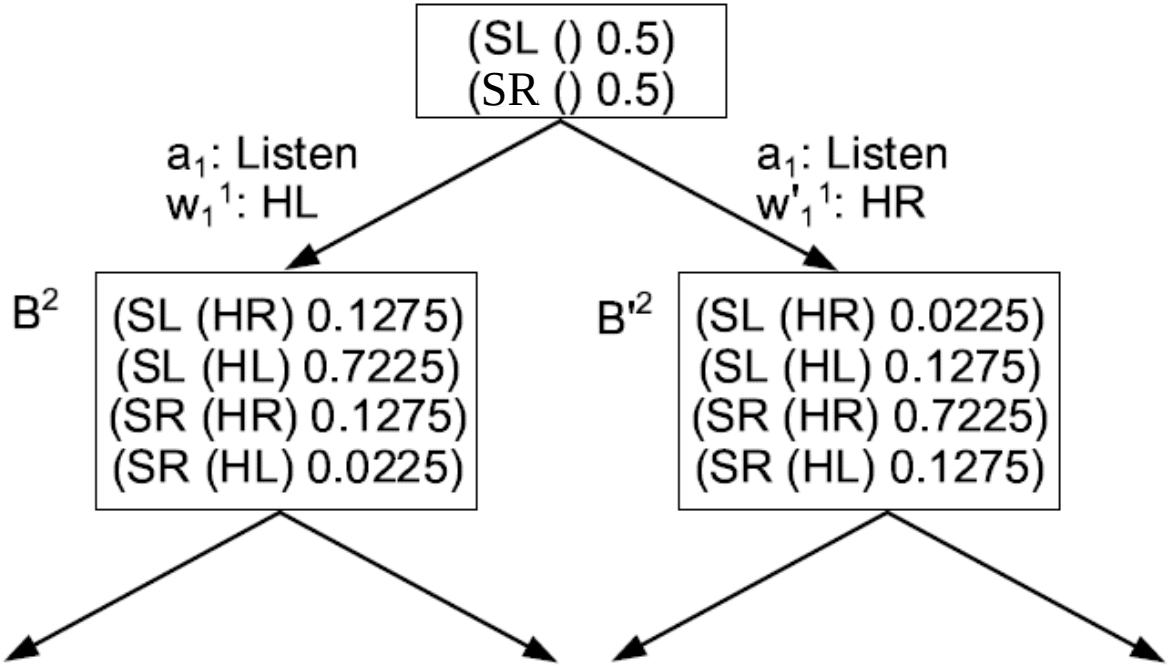
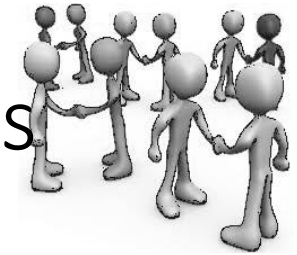


Figure 1: Trace of Tiger Scenario

Joint Equilibrium Search for Policies



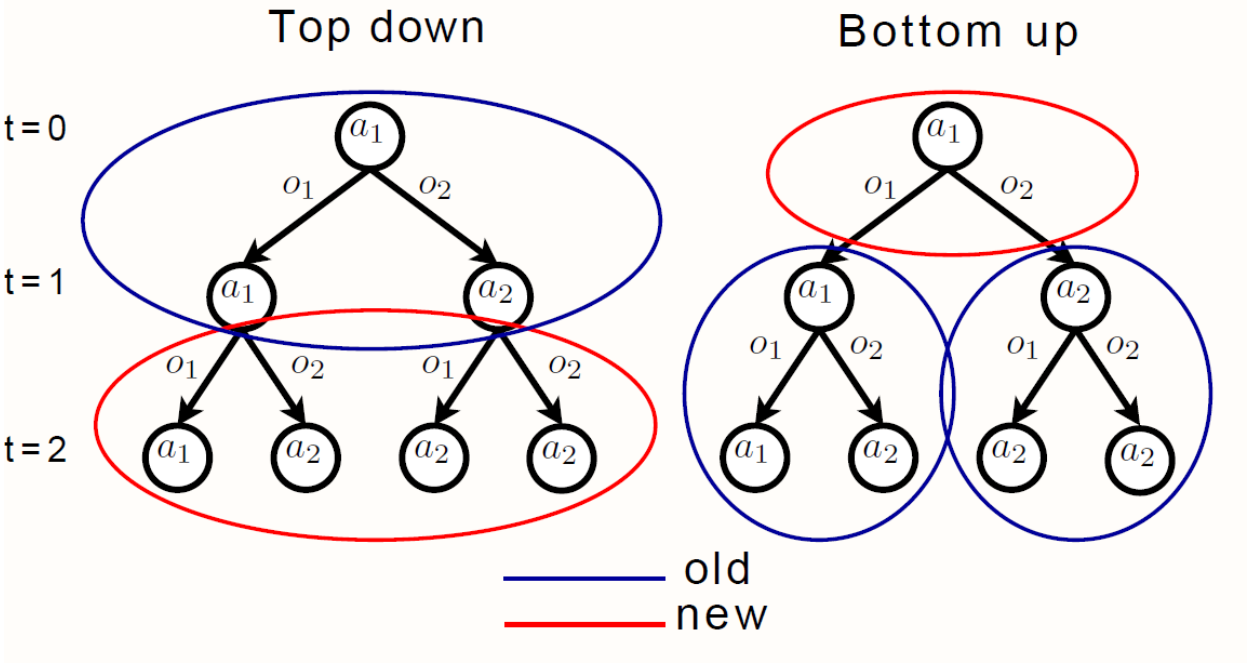
Same principles apply as for standard POMDPs

- Piecewise linear convex representation for the value function
- A similar belief propagation update as for standard POMDPs can be defined

However: The state space is growing at each time step!

- Use similar approach to point-based value iteration
- Compute reachable set of belief states from intial belief state b_0
- Thats still a lot of points
- Only compute value on this fixed set of points

Top-Down Approaches



So far: Bottom-Up: Start with policies for last time-step.
Propagate backwards in time
Alternative: Top-down: Start with first time-step, do forward search

Multi Agent A*

A* Search:

- Forward search tree
- Value at leaves is estimated by heuristic function
- Expand leave with highest return (sum of rewards in tree + heuristic)
- Admissible heuristic function: needs to overestimate return!

MA A*: top-down heuristic policy search (Szer et al., 2005).

- Requires an admissible heuristic function.
- A*-like search over partially specified joint policies:
  - Collection of policy trees for each time step

    $$\varphi_t = (\pi_0, \ldots, \pi_t)$$
  - Policy trees are defined for each agent

    $$\pi_t = (\pi_{t,1}, \ldots, \pi_{t,n})$$
  - Policy tree: Direct mapping from observation histories to action

    $$\pi_{t,i} : \mathcal{O}_i^{1:t} \rightarrow A_i$$

Multi-Agent A*

- Heuristic value for partial joint policies φ_t

$$\hat{V}(\varphi_t) = \underbrace{V_{0:t}(\varphi_t)}_{\text{expected reward for the first t time steps}} + \underbrace{\hat{V}_{t+1:T}(\varphi_t)}_{\text{Heuristic value starting from the "state" } \varphi_t}$$

- Good heuristics can avoid the construction large trees
  - If the heuristic would be accurate, the tree would be only expanded along the real optimal path

Heuristic Function

Can be defined by solving simplified problem settings:

$$\tilde{V}_{t+1:T}(\varphi_t) = \sum_{h_t \in H_t} p(h_t|b_0, \varphi_t)\tilde{Q}(h_t, \varphi_t(h_t))$$

- h_t is a history over observations and actions
- $\tilde{Q}(h_t, \varphi_t(h_t))$… Q-function of an easier (Po)MDP starting from history h_t and taking action $\varphi_t(h_t)$
- We need to average over all possible histories starting with intial belief b_0

Heuristic Function

What can we use for Q?

Q_{MDP}:

$$\tilde{Q}(h_t, a) = \sum_s p(s|h_t) Q_{\mathrm{MDP}}(s, a)$$

- Average Q-value for an MDP defined on the same system dynamics (state is observable)
- The value of the MDP is for sure better (overestimates) than the one on the Dec-POMDP
- Relatively simple to compute
- Loose bound: Assumes full observability and centralized control

Heuristic Function

What can we use for Q?

Q_{POMDP} (Szer et al., 2005; Roth et al., 2005):

- Average Q-value for an POMDP defined on the same system and observation model
- Tighter bound: Assumes centralized control
- But harder to compute

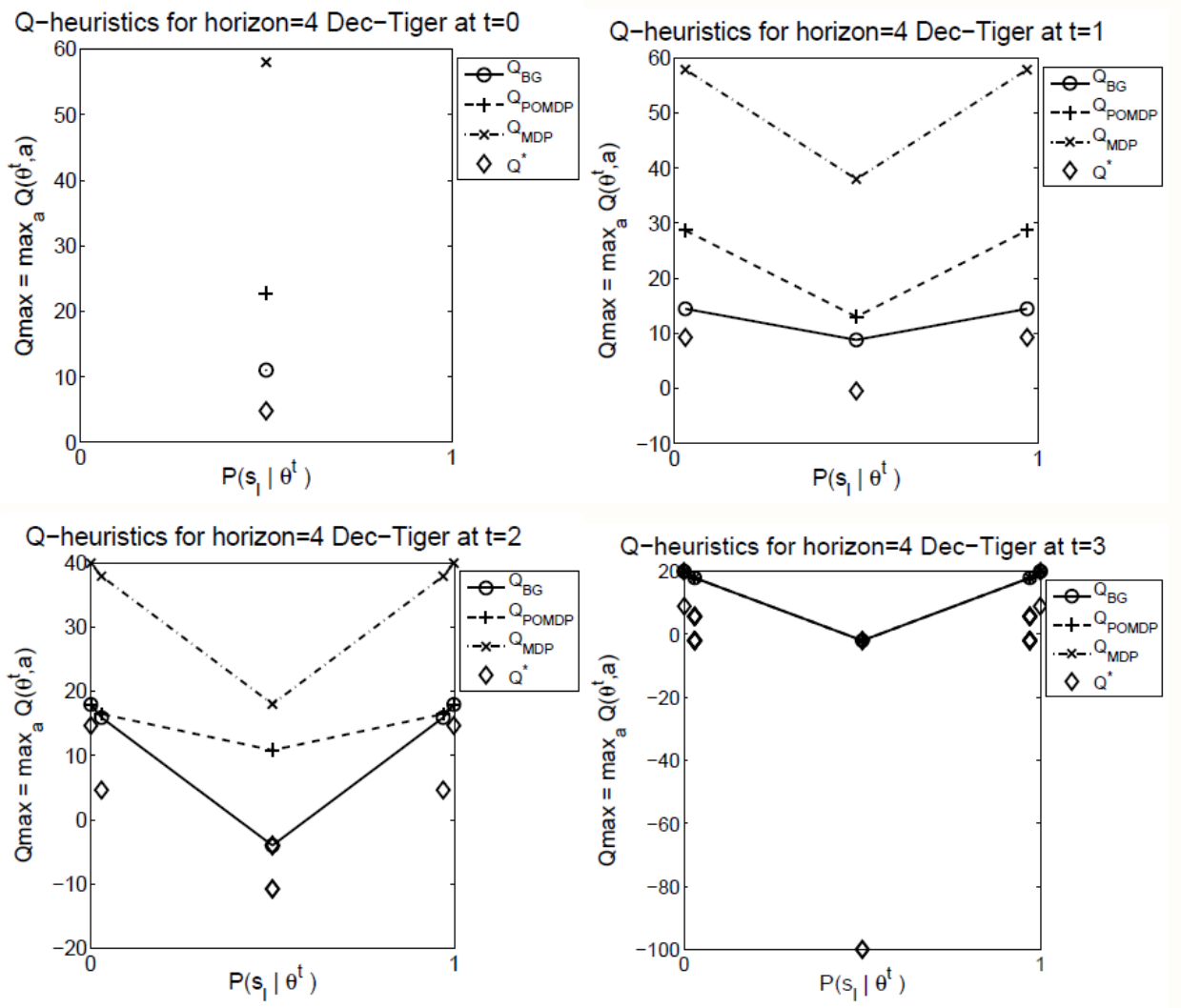
Q_{BG} (Oliehoek and Vlassis, 2007)

- assumes centralized observations, but delayed by 1 step.

It can be shown that:

$$Q^* \leq Q_{\mathrm{BG}} \leq Q_{\mathrm{POMDP}} \leq Q_{\mathrm{MDP}}$$

Illustration



Outline

Problem Formulation

Finite Horizon approaches

- Bottom-Up
- Top-Down

Infinite Horizon approaches

- Optimizing finite state controllers

Infinite Horizon Policies

A large enough horizon can be used to approximate an infinite-horizon solution, but this is neither efficient nor compact

Specialized infinite-horizon solutions have also been developed:

- Use finite state controllers as policy
- Can be optimized by different methods
- We look at a non-linear programming approach

Finite State Controllers:

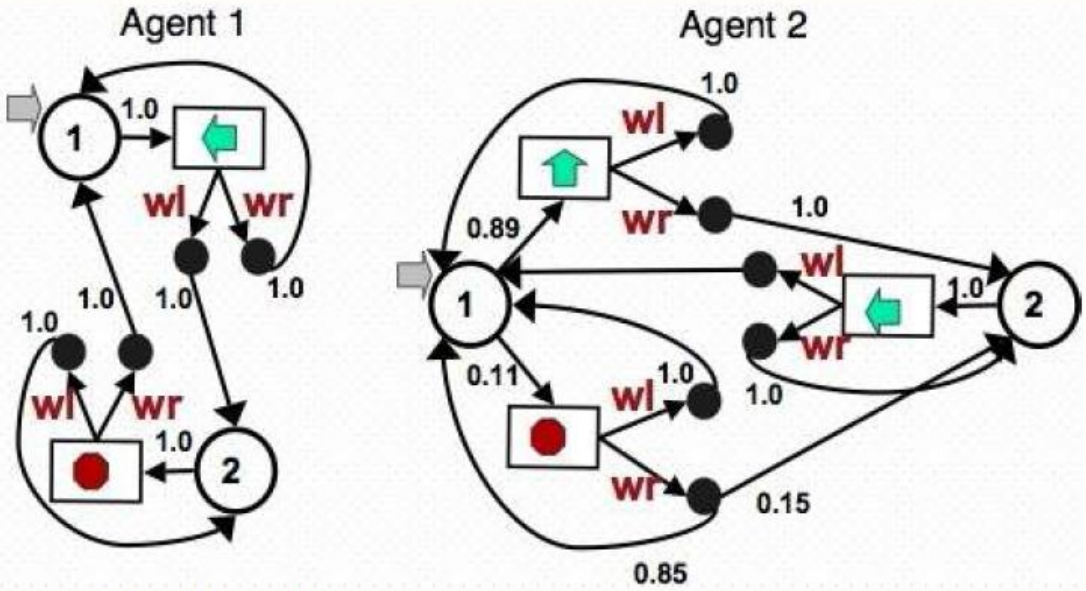
- The controller has internal nodes *q* which serve as memory
- Each node has an action policy defined $\pi_i(a|q)$
- We also define a transition probability between the notes

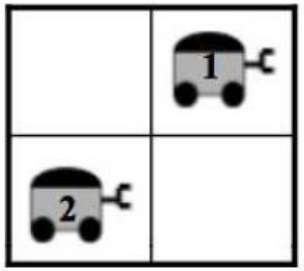
$$P_i(q'|q,o,a)$$

  - Depends on observation and action

Using controllers for infinite-horizon policies

Example: 2 Agents in a grid





- Periodic policies
- Inherently infinite-horizon
- Randomness reduces memory limitations

- Nodes define actions, $P_i(a|q)$
- Transitions based on observations seen, $P_i(q'|q,o)$

Value function for controllers

Value for the nodes $\boldsymbol{q} = (q_1, \ldots, q_n)$ of all players and state s can be found as

$$V(\boldsymbol{q}, s) = \sum_{\boldsymbol{a}} p(\boldsymbol{a}|\boldsymbol{q}) \left(r(s, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{o}, s'} p(s'|s, \boldsymbol{a}) O(\boldsymbol{o}|s') \sum_{\boldsymbol{q}'} p(\boldsymbol{q}'|\boldsymbol{q}, \boldsymbol{o}) V(\boldsymbol{q}', s') \right)$$

where:

- $p(\boldsymbol{a}|\boldsymbol{q}) = \prod_i \pi_i(a_i|q_i)$ … is the product of the policies and
- $p(\boldsymbol{q}'|\boldsymbol{q}, \boldsymbol{o}) = \prod_i p_i(q_i'|q_i, o_i)$ … is the product of the transition functions

of the controllers.

I.e., policies and transition function need to be independent.

Infinite-horizon approaches

Use a fixed-size controller

- Want best state controller for the given size
- How can the action selection and node transition parameters be set?
  - Deterministic controllers: using heuristic search methods
  - Stochastic controllers: using continuous non-linear optimization

Controller estimation by NLP

Estimating the controller parameters (policy, transition) can be written down as non-linear constraint optimization

Variables: $V(\boldsymbol{q}, s) \ldots$ value for all states and notes

$p(\boldsymbol{a}|\boldsymbol{q}) \ldots$ policies for all controllers (product)

$p(\boldsymbol{q}'|\boldsymbol{q}, \boldsymbol{o}, \boldsymbol{a}) \ldots$ transitions for all controllers (product)

Maximize expected value: $\sum_s b_0(s) V(\boldsymbol{q}_0, s)$

Given Bellman constraints:

$$V(\boldsymbol{q}, s) = \sum_{\boldsymbol{a}} p(\boldsymbol{a}|\boldsymbol{q}) \left(r(s, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{o}, s'} p(s'|s, \boldsymbol{a}) O(\boldsymbol{o}|s') \sum_{\boldsymbol{q}'} p(\boldsymbol{q}'|\boldsymbol{q}, \boldsymbol{o}) V(\boldsymbol{q}', s') \right)$$

… we are still missing… independence !

Controller estimation by NLP

For each agent i:

- Independence constraints:

$$\forall a_i, q_i, \boldsymbol{q}^f_{-i} : \underbrace{\sum_{a_{-i}} p(\boldsymbol{a}|\boldsymbol{q})}_{p(a_i|\boldsymbol{q})} = \sum_{a_{-i}} p(\boldsymbol{a}|q_i, \boldsymbol{q}^f_{-i})$$

  - The marginal $p(a_i|\boldsymbol{q})$ should only depend on q_i

$$\forall a_i, \boldsymbol{q}^f_{-i}, q_i, \boldsymbol{q}^f_{-i}, o_i, \boldsymbol{o}^f_i : \underbrace{\sum_{\boldsymbol{q}'_{-i}} p(\boldsymbol{q}'|\boldsymbol{q}, \boldsymbol{o}, \boldsymbol{a})}_{p(q_i|\boldsymbol{q},\boldsymbol{a},\boldsymbol{o})} = \sum_{\boldsymbol{q}'_{-i}} p(\boldsymbol{q}'|(q_i, \boldsymbol{q}^f_{-i}), (a_i, \boldsymbol{a}^f_{-i}), (o_i, \boldsymbol{o}^f_{-i}))$$

  - The marginal $p(q_i|\boldsymbol{q},\boldsymbol{a},\boldsymbol{o})$ should only depend on q_i, o_i and a_i.

Controller estimation by NLP

For each agent i:

- Probability constraints:

$$\forall \boldsymbol{q} : \sum_{\boldsymbol{a}} p(\boldsymbol{a}|\boldsymbol{q}) = 1, \text{and } \forall \boldsymbol{q}, \boldsymbol{a} : p(\boldsymbol{a}|\boldsymbol{q}) \geq 0$$

$$\forall \boldsymbol{a}, \boldsymbol{o}, \boldsymbol{q} : \sum_{\boldsymbol{q}'} p(\boldsymbol{q}'|\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{o}) = 1, \text{and } \forall \boldsymbol{q}, \boldsymbol{a}, \boldsymbol{o}, \boldsymbol{q}' : p(\boldsymbol{q}'|\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{o}) \geq 0$$

The resulting optimization problem is:

- Non-linear
- Huge for already quite small problems (number of constraints)
- Still, one of the most elegants solutions so far
- Can be solved by non-linear constraint optimizers, e.g. fmincon

Dec-POMDP Benchmarks

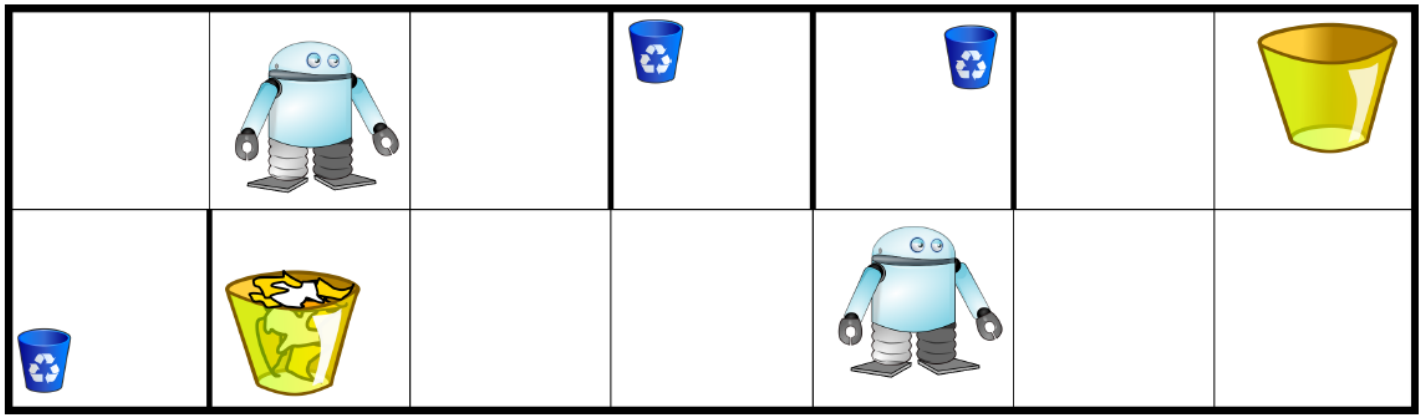
- Dec-Tiger
- Recycling Robots
- Cooperative Box-Pushing
- Mars Rover
- Fire Fighter Problem

All of these problems are rather simple benchmark tasks

Recycling Robots

2 Robots:

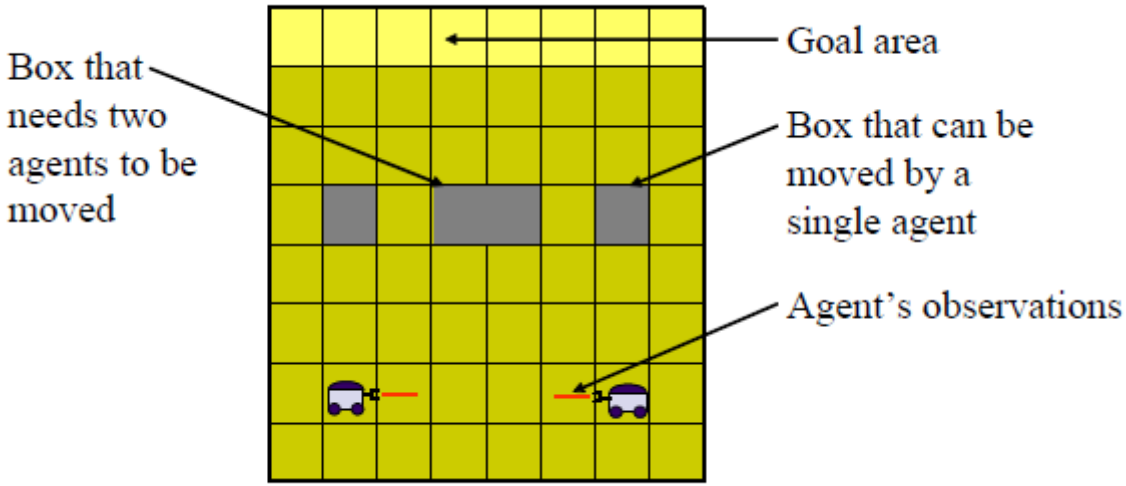
- Need to empty trash
- Big trash needs 2 robots
- Robots can only see status of neighbored trash



Cooperative Box-Pushing

2 agents:

- Need to push all boxes in the goal area
- Big box requires 2 robots to be pushed
- Local observations: only observe field in front of the robot
- State: position of the 2 agents and 3 boxes



Results

Cooperative Box Pushing: 100 states, 4 actions, 5 obs

	Value					Time				
Size	NLP	NLP fix	Mealy NLP	DEC-BPI	BFS	NLP	NLP fix	Mealy NLP	DEC-BPI	BFS
1	-1.58	n/a	123.46	-10.37	-2	20	n/a	12	26	1696
2	31.97	-6.25	124.20	3.29	–	115	18	31	579	–
3	46.28	5.10	133.67	9.44	–	683	27	217	4094	–
4	50.64	18.78	143.14	7.89	–	5176	44	774	11324	–
5	–	53.13	–	14.76	–	–	92	–	27492	–
6	–	73.25	–	–	–	–	143	–	–	–
7	–	80.47	–	–	–	–	256	–	–	–

Conclusion

- Dec-POMDPs are a relatively new research field
- Optimal solutions are intractable
- Policies can be obtained by bottom-up search or top-down search over policy trees
- Finite-State controllers can represent controllers for infinite horizon problems
- Benchmark problems are still toyish