MODELS OF COORDINATION IN MULTIAGENT DECISION MAKING

Chris Amato

Many slides from a larger tutorial with Matthijs Spaan and Shlomo Zilberstein

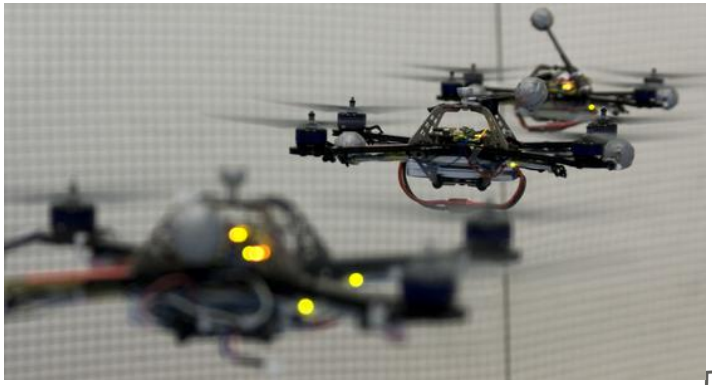
Some other slides courtesy of: Dan Bernstein, Frans Oliehoek and Alan Carlin

Massachusetts Institute of Technology

CSAIL

General overview

- Agents situated in a world, receiving information and choosing actions
  - Uncertainty about outcomes and sensors
  - Sequential domains
  - Cooperative multi-agent
  - Decision-theoretic approach
- Developing approaches to scale to real-world domains

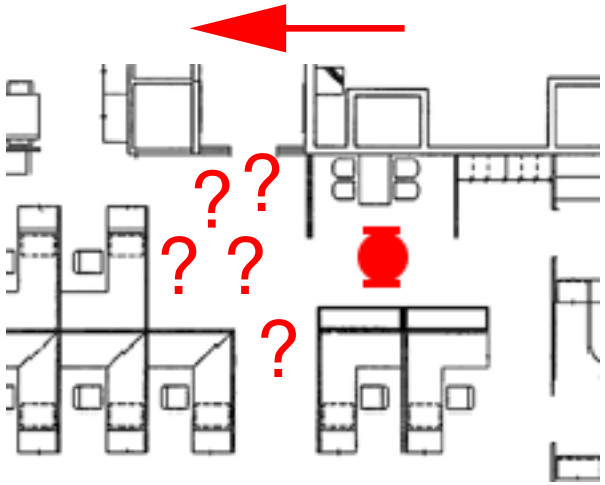


Outline

- Background on decision-theoretic planning
- Models for cooperative sequential decision-making
  - Dec-POMDPs and MTDPs
  - Notable subclasses (Dec-MDPs, ND-POMDPs)
  - Related competitive models (POSGs, I-POMDPs)
- Optimal algorithms
  - Finite and infinite horizon
  - Top down and bottom up
- Approximate algorithms
  - Finite and infinite horizon
- Communication
- Applications being targeted
- Conclusion

Decision-theoretic planning

- Tackles uncertainty in sensing and acting in a principled way
- Popular for single-agent planning under uncertainty (MDPs, POMDPs)
- We need to model:
  - Each agent's actions
  - Their sensors
  - Their environment
  - Their task



Modeling assumptions

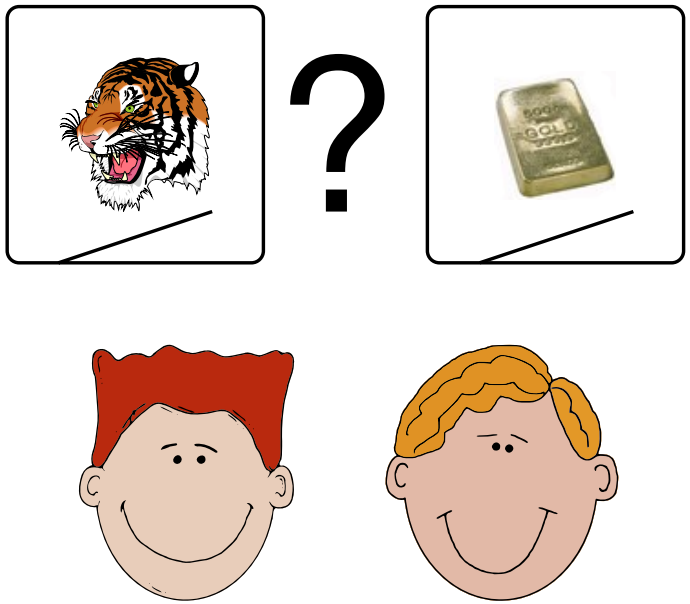
- Sequential decisions: problems are formulated as a sequence of discrete “independent” decisions
- Markovian environment: the state at time *t* depends only on the events at time *t-1*
- Stochastic models: the uncertainty about the outcome of actions and sensing can be accurately captured
- Objective encoding: the overall objective can be encoded using cumulative (discounted) rewards over time steps

Problem aspects

- On-line vs. off-line
- Centralized vs. distributed
  - Planning
  - Execution
- Cooperative vs. self-interested
- Observability
- Communication

A toy example (decentralized tiger)

- 2 agents with 2 doors
- A tiger is behind one and a treasure is behind the other
- Can listen for a noisy signal about the location of the tiger
- If either agent opens the door with the tiger, they both get mauled
- If the door with treasure is opened, they share the reward
- Don't know each other's actions and observations



Single agent/centralized models

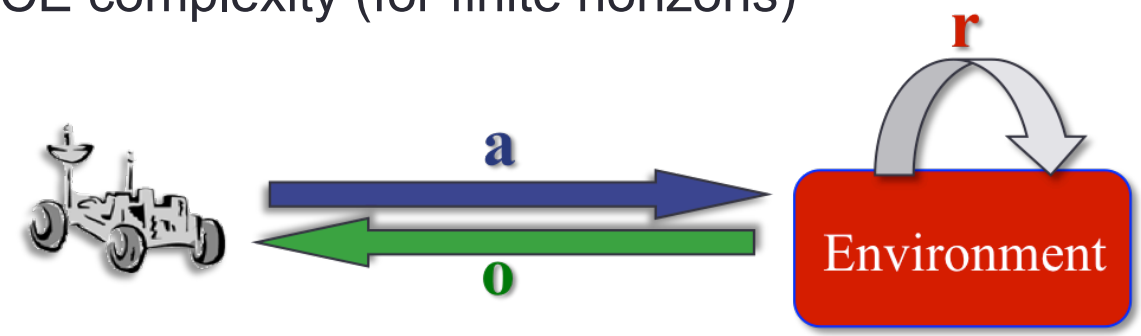
- Markov decision processes (MDPs)
  - Stochastic actions, but fully observe state at each step
  - P complexity

- Maximize value $V(s) = R(s,a) + \gamma \max_a \sum_{s'} P(s' \mid s,a) V(s')$

- Can be multiagent, but centralized execution (and large size)

Single agent/centralized models

- Partially observable Markov decision processes (POMDPs)
  - Receive an observation rather than true state
  - PSPACE complexity (for finite horizons)



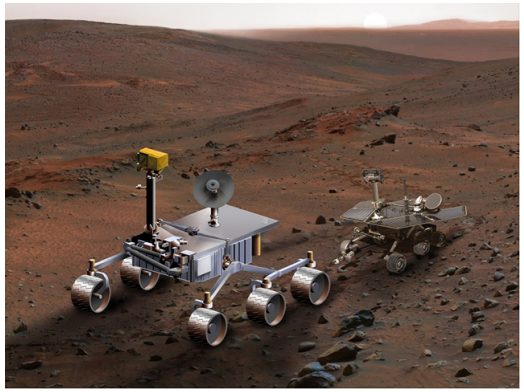
- Maximize value in a similar way (but over distributions over states or *beliefs*)
- Can also be (centralized) multiagent

Decentralized domains

- Cooperative multiagent problems
- Each agent's choice affects all others, but must be made using only local information
- Properties
  - Often a decentralized solution is required
  - Multiple agents making choices independently of the others
  - Does not require communication on each step (may be impossible or too costly)
  - But now agents must also reason about the previous and future choices of the others (more difficult)

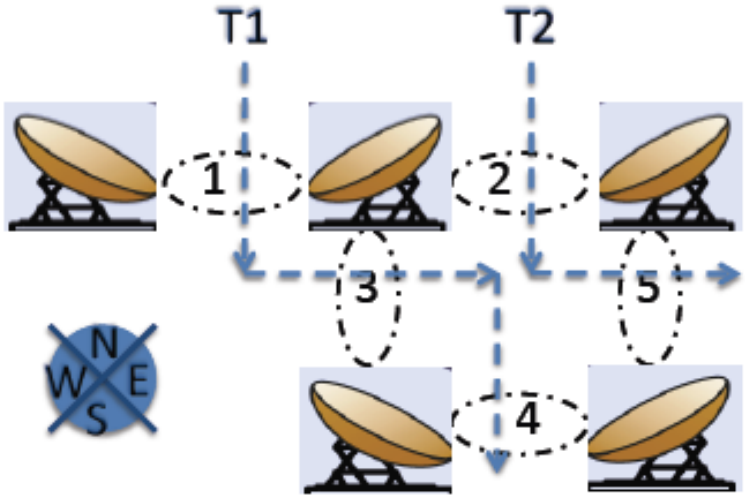
Example cooperative multiagent problems

- Multi-agent planning (examples below, reconnaissance, etc.)
- Human-robot coordination (combat, industry, etc.)
- Sensor networks (e.g. target tracking from multiple viewpoints)
- E-commerce (e.g. decentralized web agents, stock markets)



Sensor network problems

- Sensor networks for
  - Target tracking (Nair et al., 05, Kumar and Zilberstein 09-AAMAS)
  - Weather phenomena (Kumar and Zilberstein 09-IJCAI)
- Two or more cooperating sensors

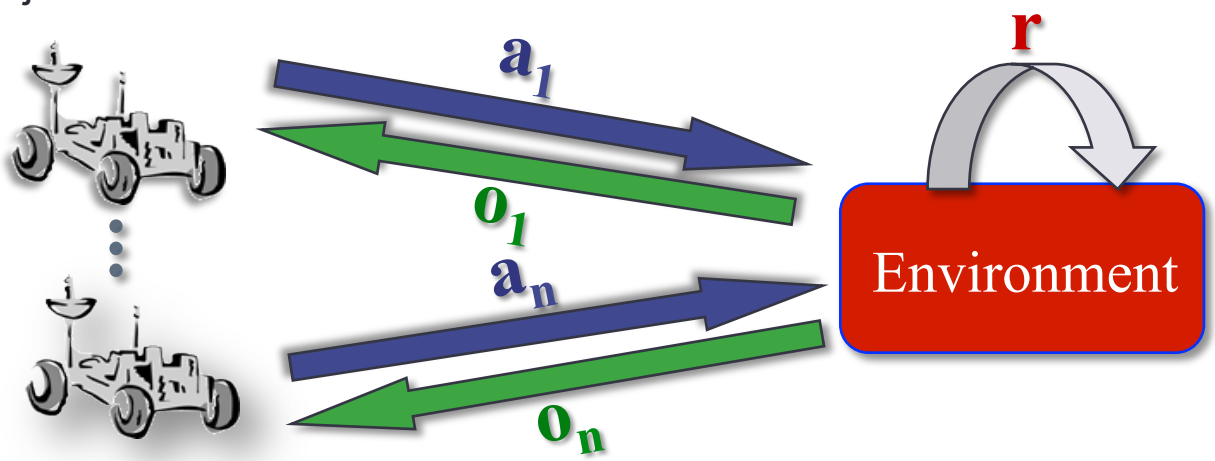


Other possible application domains

- Multi-robot coordination
  - Space exploration rovers (Zilberstein et al., 02)
  - Helicopter flights (Pynadath and Tambe, 02)
  - Navigation (Emery-Montemerlo et al., 05; Spaan and Melo 08)
- Load balancing for decentralized queues (Cogill et al., 04)
- Multi-access broadcast channels *(Ooi and Wornell, 96)*
- Network routing (Peshkin and Savova, 02)
- Sensor network management (Nair et al., 05)

Multiple cooperating agents

- Decentralized partially observable Markov decision process (Dec-POMDP) also called multiagent team decision problem (MTDP)
- Extension of the single agent POMDP
- Multiagent sequential decision-making under uncertainty
  - At each stage, each agent takes an action and receives:
    - A local observation
    - A joint immediate reward



Dec-POMDP definition

- A Dec-POMDP can be defined with the tuple:

M = <I, S, {A_i}, P, R, {Ω_i}, O>

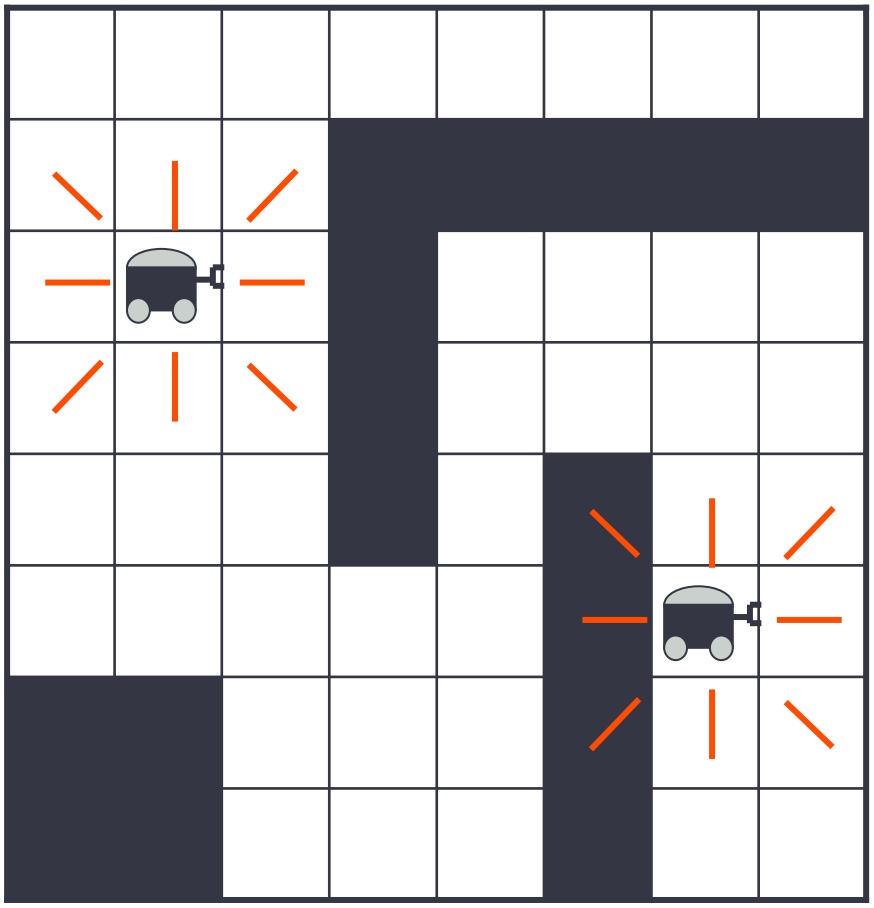
- I, a finite set of agents
- S, a finite set of states with designated initial state distribution b_0
- A_i, each agent's finite set of actions
- P, the state transition model: $P(s' \mid s, \vec{a})$
- R, the reward model: $R(s, \vec{a})$
- Ω_i, each agent's finite set of observations
- O, the observation model: $P(\vec{o} \mid s, \vec{a})$

Note: Functions depend on all agents

Dec-POMDP solutions

- A local policy for each agent is a mapping from its observation sequences to actions, $\Omega^* \rightarrow A$
  - State is unknown, so beneficial to remember history
- A joint policy is a local policy for each agent
- Goal is to maximize expected cumulative reward over a finite or infinite horizon
  - For infinite-horizon cannot remember the full observation history
  - In infinite case, a discount factor, γ, is used

Example: 2-Agent Navigation



States: grid cell pairs

Actions: move ⇧, ⇩, ⇨, ⇦, stay

Transitions: noisy

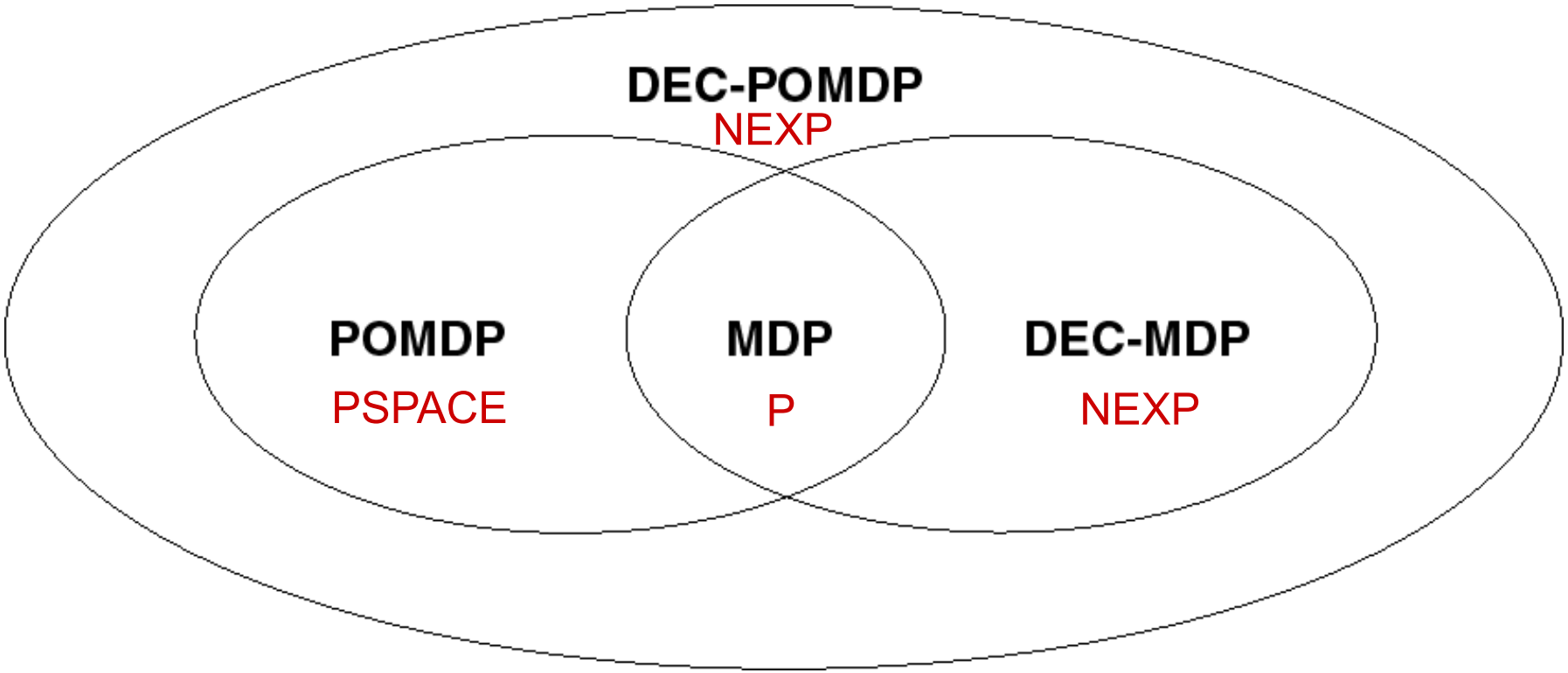
Observations: red lines

Rewards: negative unless sharing the same square

Challenges in solving Dec-POMDPs

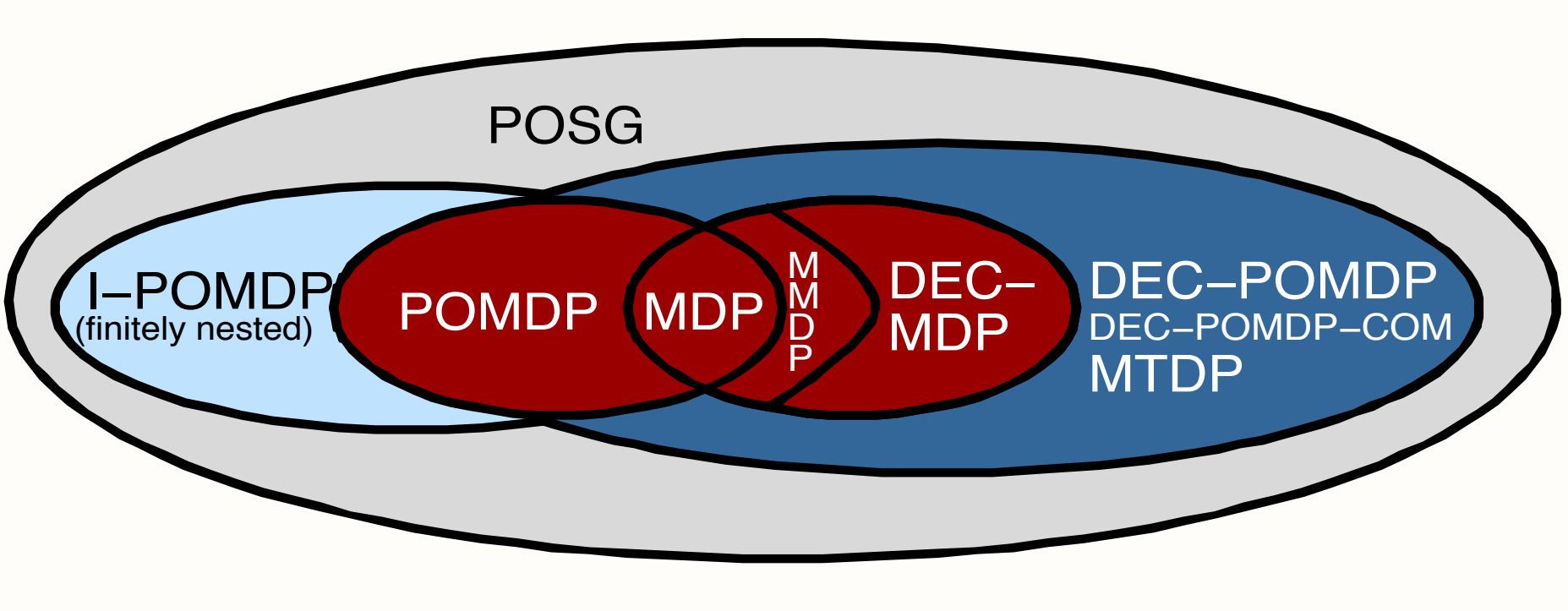
- Partial observability makes the problem difficult to solve
- No common state estimate (centralized belief state)
  - Each agent depends on the others
  - This requires a belief over the possible policies of the other agents
  - Can't transform Dec-POMDPs into a continuous state MDP (how POMDPs are typically solved)
- Therefore, Dec-POMDPs cannot be solved by centralized (POMDP) algorithms

General complexity results



subclasses and finite horizon complexity results

Relationship with other models



Ovals represent complexity, while colors represent number of agents and cooperative or competitive models

POSGs

- A partially observable stochastic game (POSG) is a tuple M = <I, S, $\{A_i\}$, P, $\{R_i\}$, $\{\Omega_i\}$, O> where
  - All the components except the reward function are the same as in a DEC-POMDP
  - Each agent has an individual reward function: $R_i : A_i \times S \rightarrow \Re$
    - $R_i(s,a_i)$ denotes the reward obtained after action a_i was taken by agent i and a state transition to s' occurred
- This is the self-interested version of the DEC-POMDP model

I-POMDPs

- Interactive POMDPs (I-POMDPs) extend state space with behavioral models of other agents (Gmytrasiewicz and Doshi 05)
- Agents maintain beliefs over physical and models of others
- Recursive modeling
- When assuming a finite nesting, beliefs and value functions can be computed (approximately).
- Finitely nested I-POMDPs can be solved as a set of POMDPs

Dec-MDPs

- *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 $O(\vec{o} \mid s', \vec{a}) > 0$ then $P(s' \mid \vec{o}) = 1$
- A *decentralized Markov decision process (Dec-MDP)* is a Dec-POMDP with joint full observability.
- A stronger result: The problem is NEXP-hard even when the state is jointly observed!
  - That is, two-agent finite-horizon DEC-MDPs are NEXP-hard.

Classes of Dec-POMDPs

- A *factored n-agent Dec-MDP* is a Dec-MDP for which the world state can be factored into *n+1* components, $S = S_0 \times S_1 \times \ldots \times S_n$
- A factored, *n*-agent Dec-MDP is said to be *locally fully observable* if each agent observes its own state component

$$\forall o_i \exists \hat{s}_i : P(\hat{s}_i \mid o_i) = 1$$

- Local state/observation/action $\hat{s}_i \in S_i \times S_0$ is referred to as the local state, $a_i \in A_i$ as the local action, and $o_i \in O_i$ as the local observation for agent *I*
- *Full observability = individual observability*
- A Dec-POMDP is fully observable if there exists a mapping for each agent *i*, $f_i : \Omega_i \rightarrow S$ such that whenever $O(\vec{o} \mid s', \vec{a})$ is non-zero then $f_i(o_i) = s'$

Classes of Dec-POMDPs

- A *multi-agent Markov decision process (MMDP)* is a Dec-POMDP with full observability (Boutilier, 96)
- A factored, *n*-agent Dec-MDP is said to be *transition independent* if there exists P_0 through P_n such that

$$P(s'_i \mid (s_o,\ldots,s_n),\vec{a},(s'_o,\ldots,s'_{i-1},s'_{i+1},\ldots,s'_n)) = \begin{cases} P_0(s'_0 \mid s_0) & i = 0 \\ P_i(s'_i \mid s_i,a,s'_0) & 1 \le i \le n \end{cases}$$

- A factored, *n*-agent Dec-MDP is said to be *observation independent* if there exists O_1 through O_n such that:

$$P(o_i \mid,\vec{a},(s'_o,\ldots,s'_n),(o_o,\ldots,o_{i-1},o_{i+1},\ldots,o_n)) = P(o_i \mid a_i,s'_i)$$

Classes of Dec-POMDPs

- A factored, *n*-agent Dec-MDP is said to be *reward independent* if there exist f and R_1 through R_n such that

$$R((s_0,\ldots,s_n),\vec{a}) = f(R_1(s_1,a_1),\ldots,R_n(s_n,a_n))$$

and

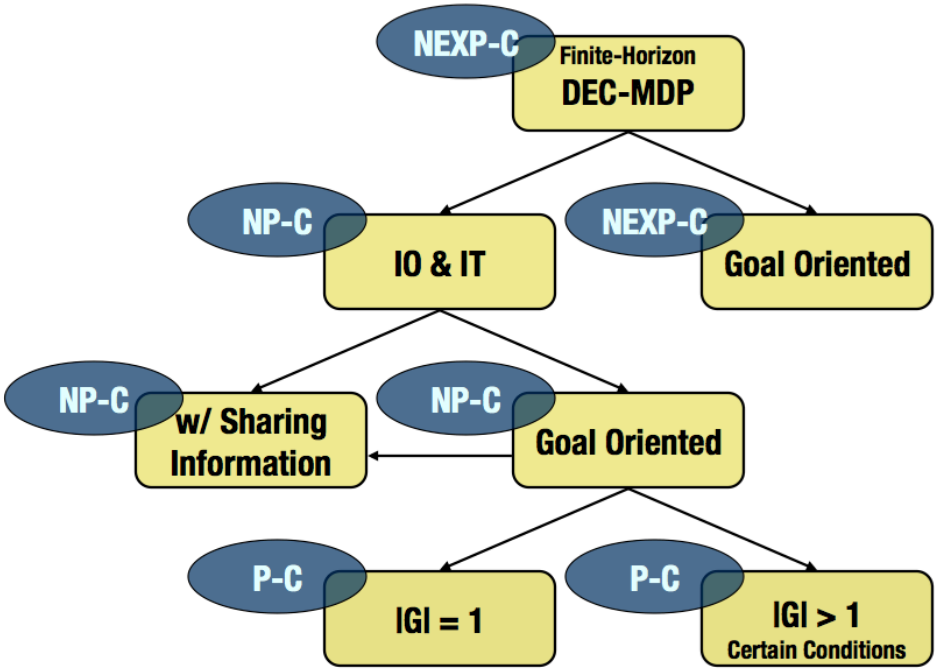
$$R_i(s_i,a_i) \leq R_i(s'_i,a'_i) \Leftrightarrow$$

$$f(R_1\ldots R_i(s_i,a_i)\ldots R_n) \leq f(R_1\ldots R_i(s'_i,a'_i)\ldots R_n)$$

- For example, additive rewards
- If a Dec-MDP has independent observations and transitions, then the Dec-MDP is locally fully observable.

Some complexity results

Observability	General communication	Free communication
Full	MMDP (P-complete)	MMDP (P-complete)
Joint full	Dec-MDP (NEXP)	MMDP (P-complete)
Partial	Dec-POMDP (NEXP)	MPOMDP (PSPACE)



Massachusetts Institute of Technology

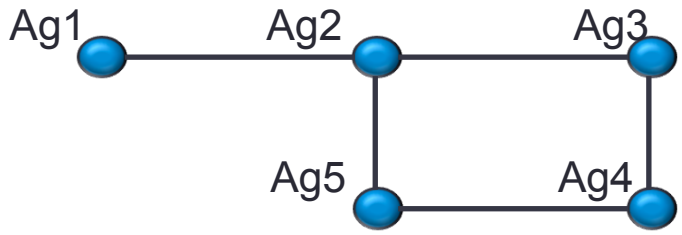
CSAIL

ND-POMDPs (Nair et al., 05)

- A *network distributed POMDP* (ND-POMDP) is a factored *n*-agent Dec-POMDP with independent transitions and observations as well as a decomposable reward function
- That is, $R((s_0, s_1, \ldots, s_n), \vec{a}) = \sum_l R_l(s_0, s_{l1}, .., s_{lk}, a_{1k}, \ldots, a_{lk})$

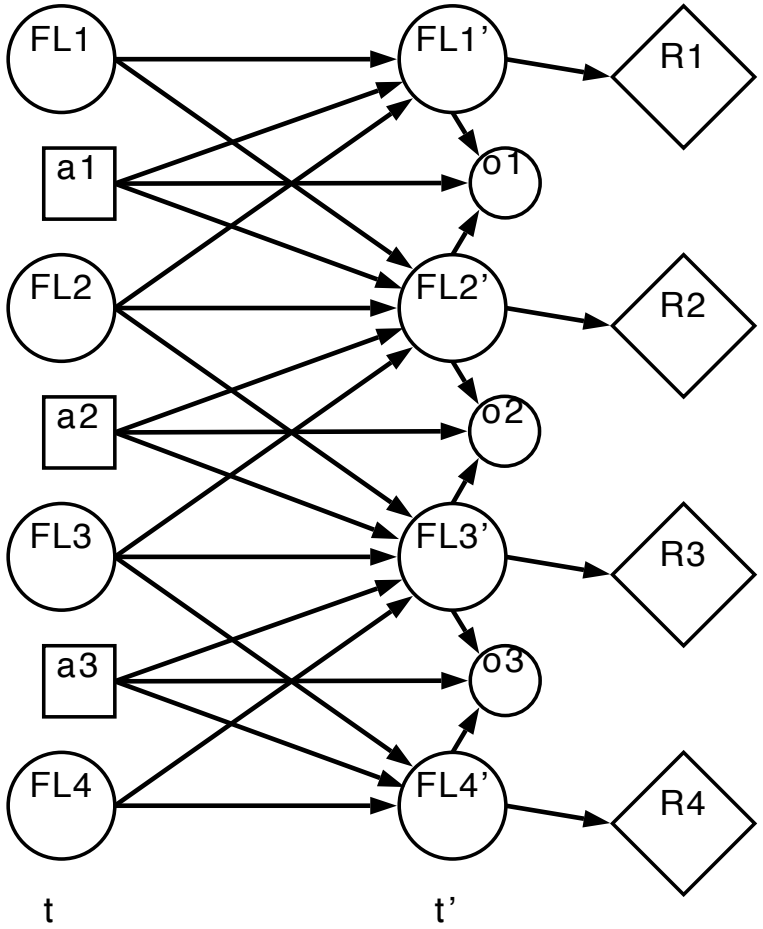
Where *l* is the subgroup of agents of size *k*

- Model locality of interaction
- Doesn't make joint full observability assumption of Dec-MDPs
- Complexity depends on *k* (but still NEXP in worst case)



Factored Dec-POMDPs (Oliehoek et al., 08)

- Motivation: exploit locality of interaction, but no strict independence assumptions
- More general and powerful than ND-POMDPs
- Less scalable (non-stationary interaction graph)

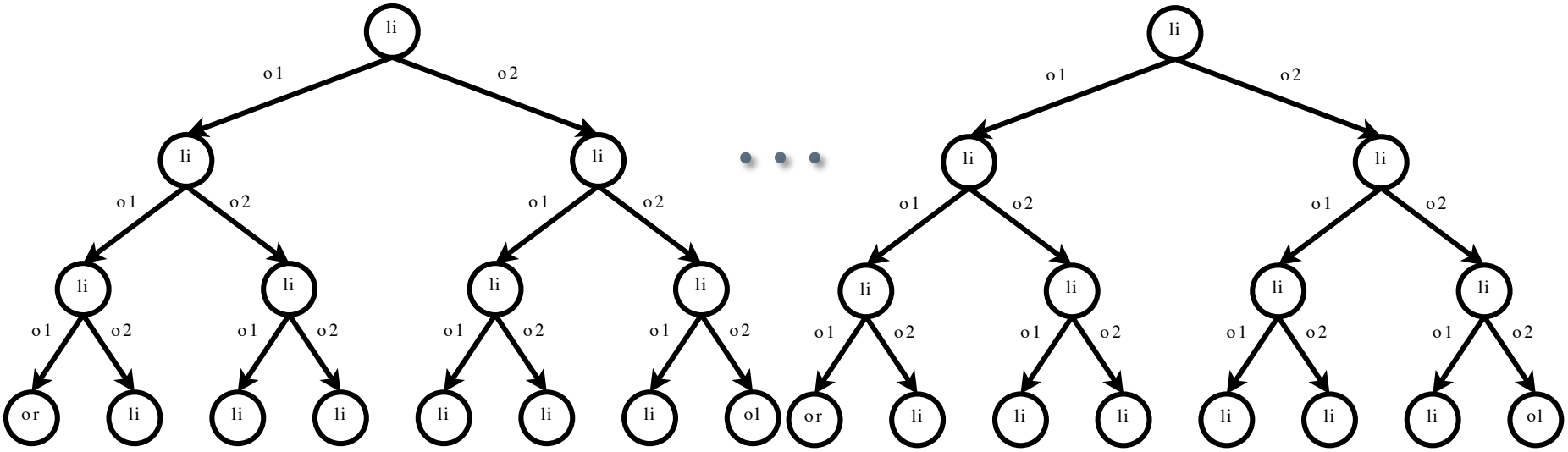


Algorithms

- How do we produce a solution for these models?
- Will discuss
  - Solution representations
  - Optimal methods
  - Approximate methods
  - Subclasses
  - Communication

Policy representations – finite horizon

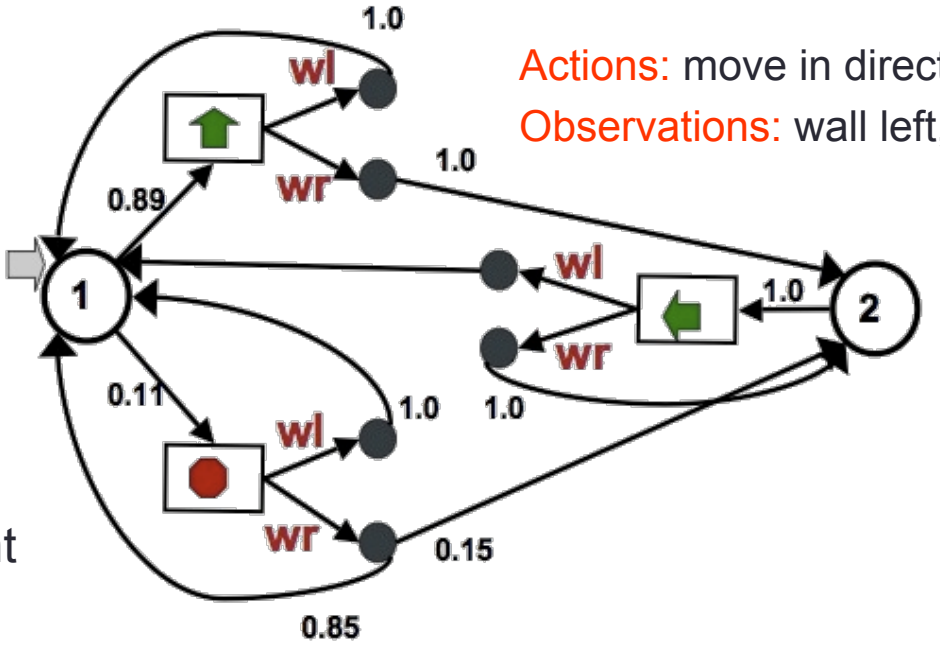
- Policy trees, one for each agent
- Nodes define actions, links define observations
- One leaf for each history for the given horizon



- Evaluation: $V(\vec{q},s) = R(s,\vec{a}_{\vec{q}}) + \gamma \sum_{s'} P(s' \mid s,\vec{a}_{\vec{q}}) \sum_{\vec{o}} O(\vec{o} \mid s',\vec{a}) V(\vec{q}_{\vec{o}},s')$
  - Starting from a node q, taking the associated action and transition

Policy representations – infinite horizon

- Designated initial node
- Nodes define actions
- Transitions based on observations seen
- Inherently infinite-horizon
- With fixed memory, randomness can help
- One controller for each agent



Actions: move in direction or stop
Observations: wall left, wall right

Action selection, $P(a|q)$: $Q \rightarrow \Delta A$
Transitions, $P(q'|q,o)$: $Q \times O \rightarrow \Delta Q$

- Value for node $\vec{q}$ and state s:

$$V(\vec{q},s) = \sum_a P(\vec{a} \mid \vec{q}) \left[R(s,\vec{a}) + \gamma \sum_{s'} P(s' \mid s,\vec{a}) \sum_o O(\vec{o} \mid s',\vec{a}) \sum_{q'} P(\vec{q}' \mid \vec{q},o) V(\vec{q}',s') \right]$$

Massachusetts Institute of Technology

CSAIL

Optimal methods

- Want to produce the optimal solution for a problem
- That is, optimal horizon *h* tree or ε-optimal controller
- Do this in a bottom up or a top down manner

Bottom up methods

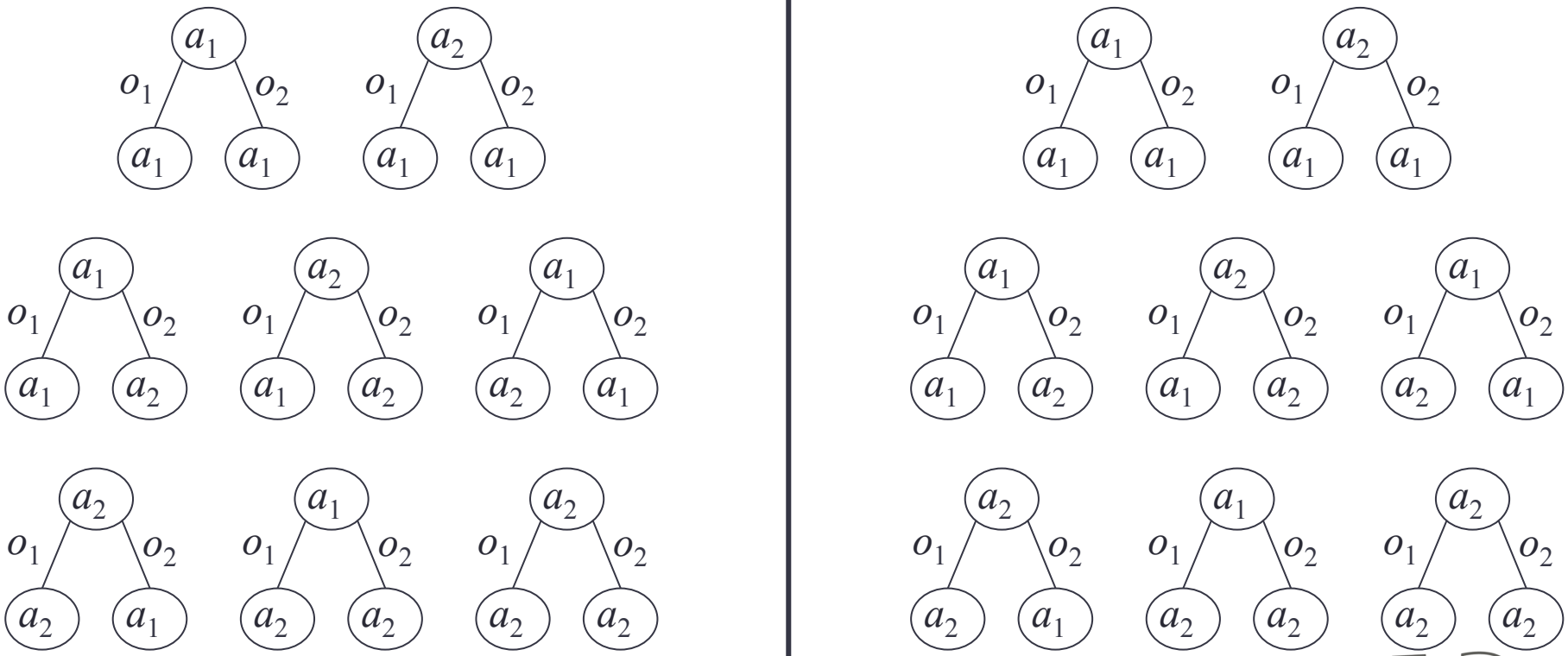
- Build the policies up for each agent simultaneously
- Begin on the last step (single action) and continue until the first
- When done, choose highest value set of trees for any initial state

Exhaustive search

- Construct all possible policies for the set of agents
- Do this in a bottom up fashion
- Stop when the desired horizon is reached
- Trivially includes an optimal set of trees when finished
- Choose the set of policies with the highest value at the initial belief

Exhaustive search example (2 agents)

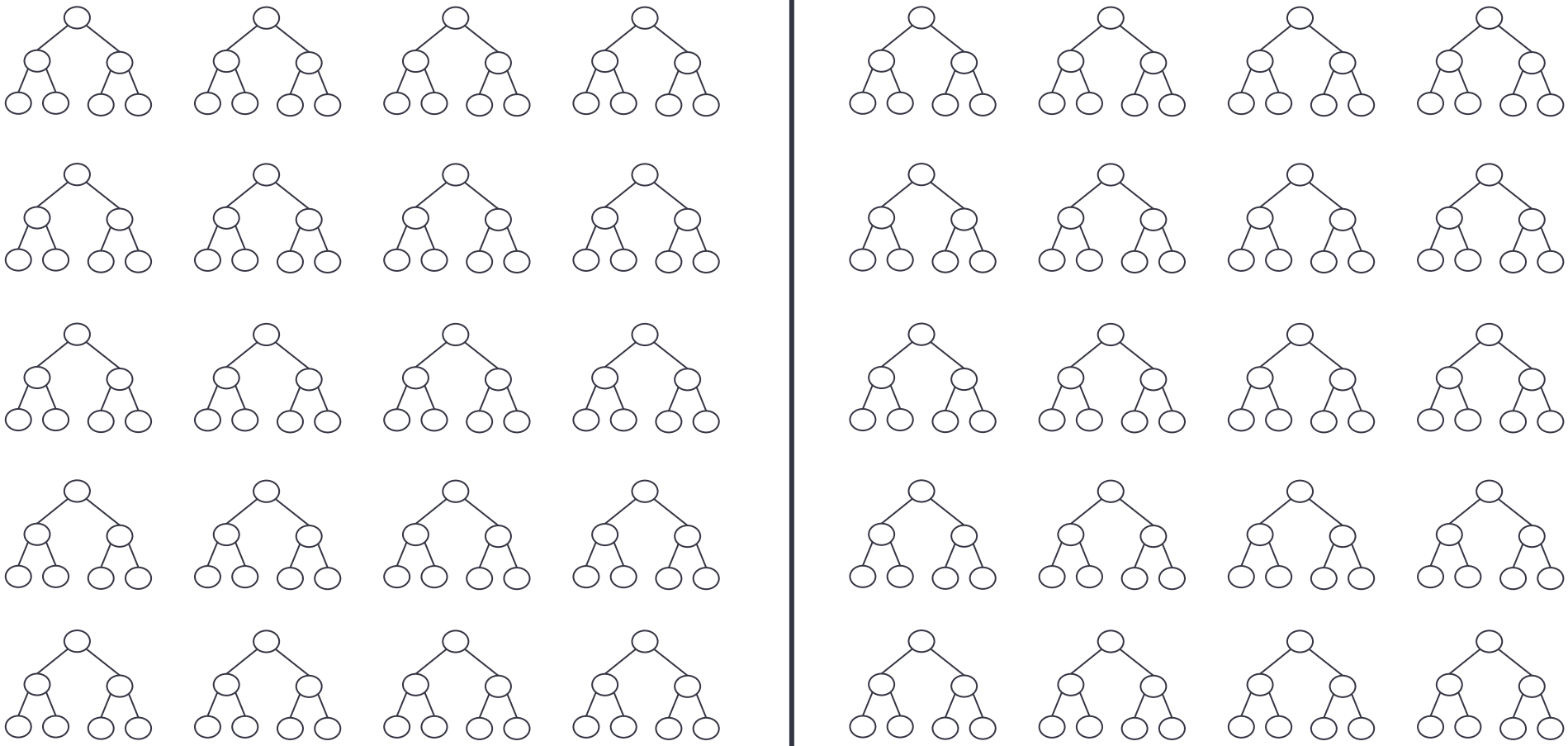
Exhaustive search continued



Massachusetts Institute of Technology

CSAIL

Exhaustive search continued



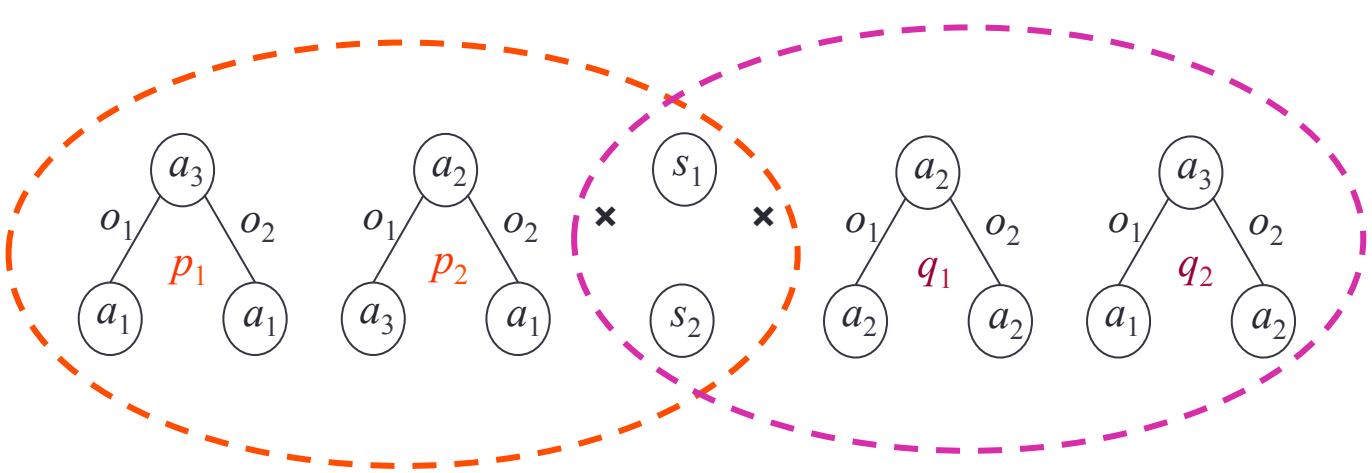
Exhaustive search summary

- 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?

Finite horizon dynamic programming (DP)

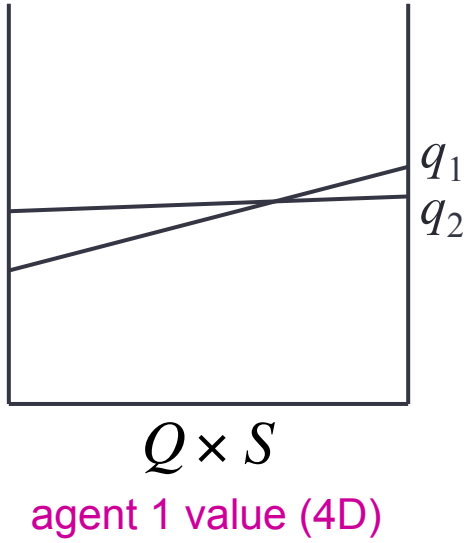
(Hansen et al. 2004)

- Build policy tree sets simultaneously
- Returns optimal policy for any initial state
- Prune using a generalized belief space (with linear program)
- This becomes iterative elimination of dominated strategies in POSGs
- For two agents:



agent 2 state space

agent 1 state space



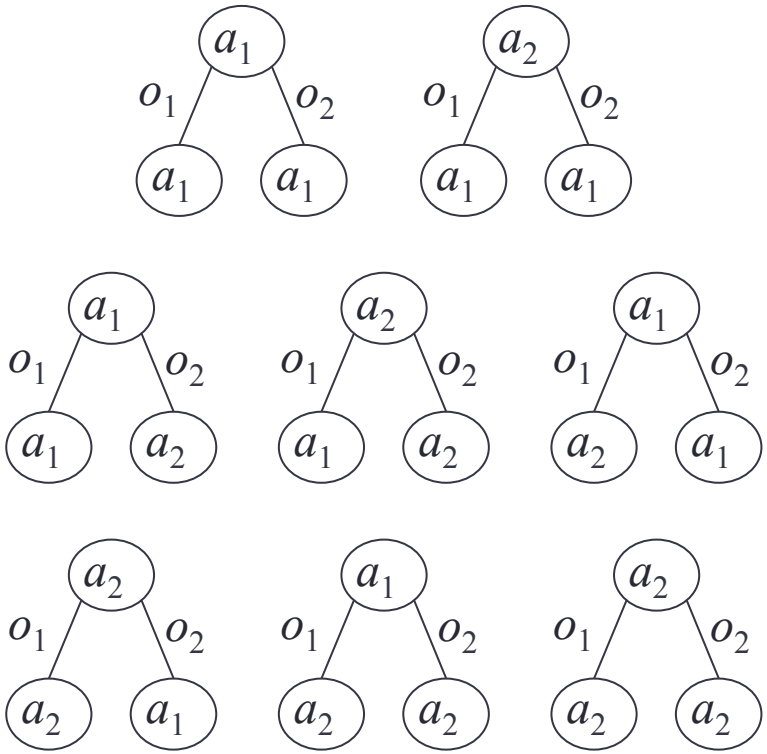
agent 1 value (4D)

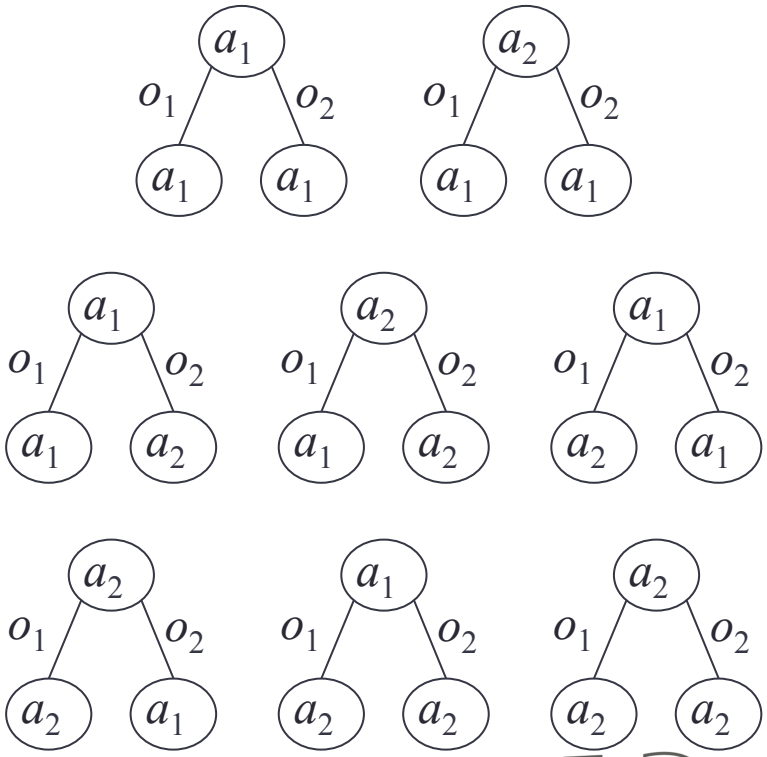
Massachusetts Institute of Technology

CSAIL

DP for DEC-POMDPs example (2 agents)

DP for DEC-POMDPs continued

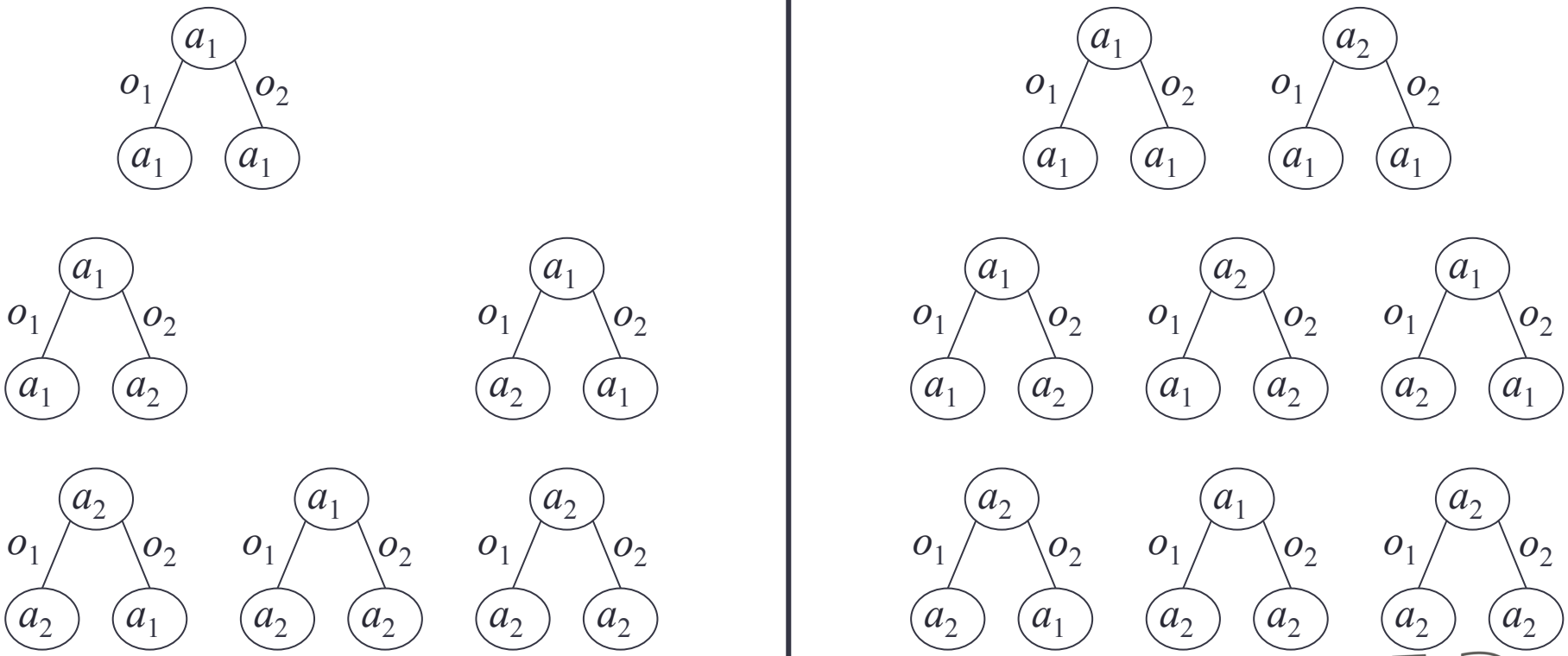




Massachusetts Institute of Technology

CSAIL

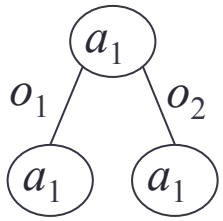
DP for DEC-POMDPs continued

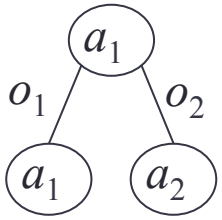


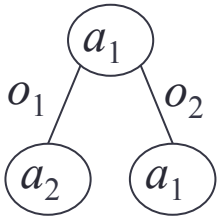
Massachusetts Institute of Technology

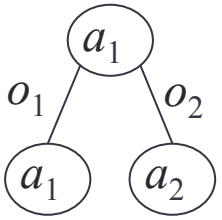
CSAIL

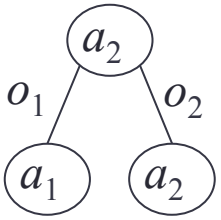
DP for DEC-POMDPs continued

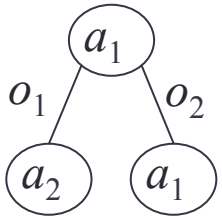


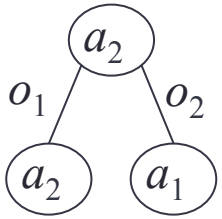


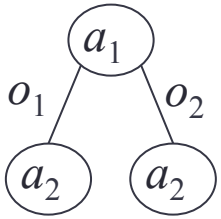


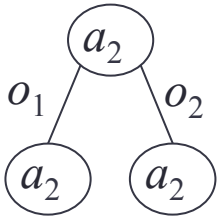


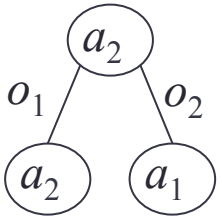


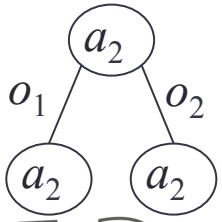








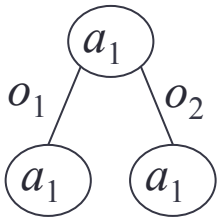


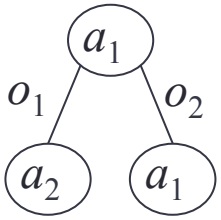


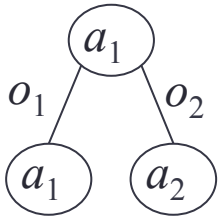
Massachusetts Institute of Technology

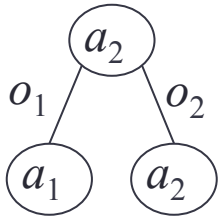
CSAIL

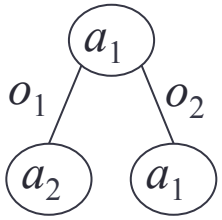
DP for DEC-POMDPs continued

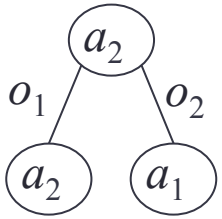


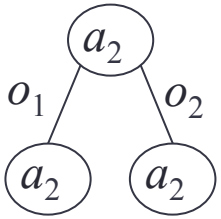


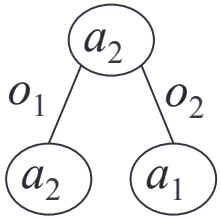


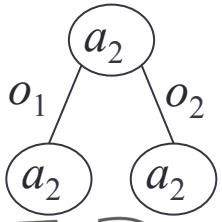








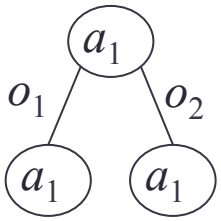


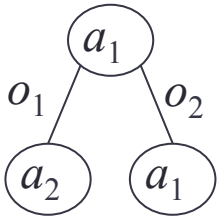


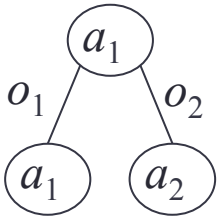
Massachusetts Institute of Technology

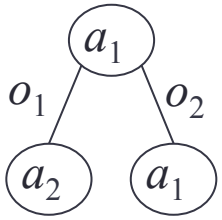
CSAIL

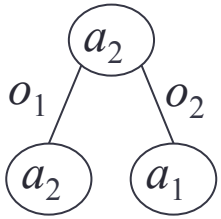
DP for DEC-POMDPs continued

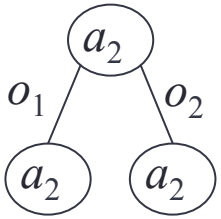


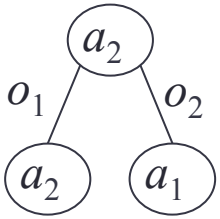


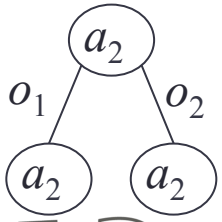








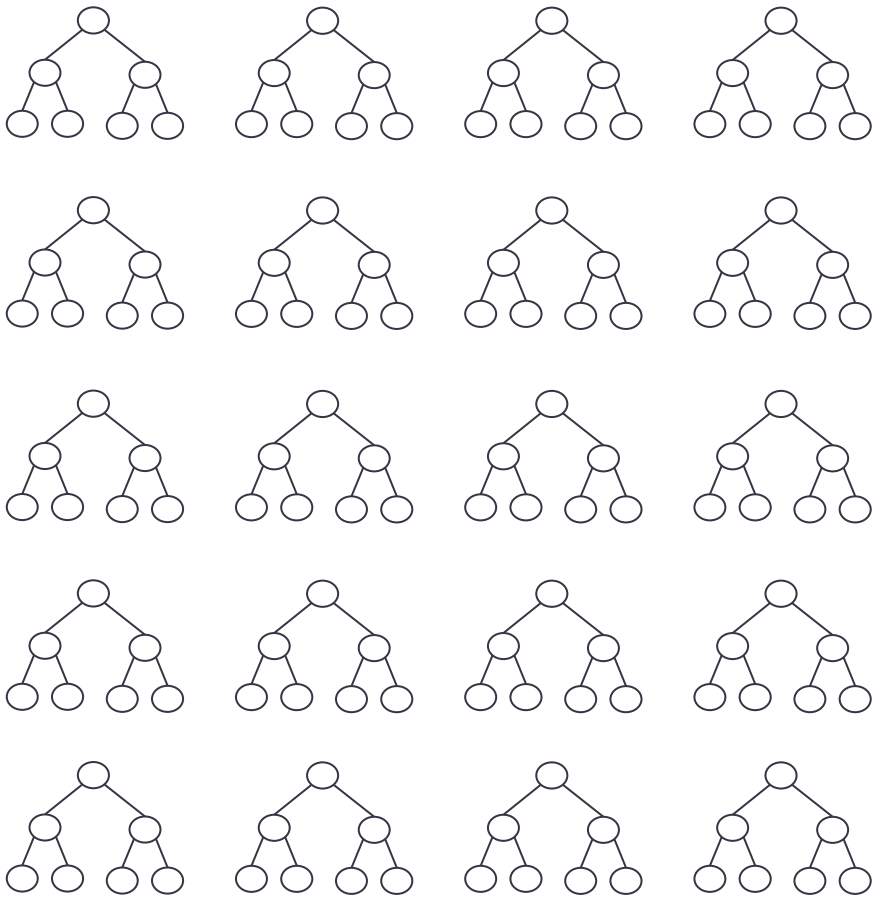


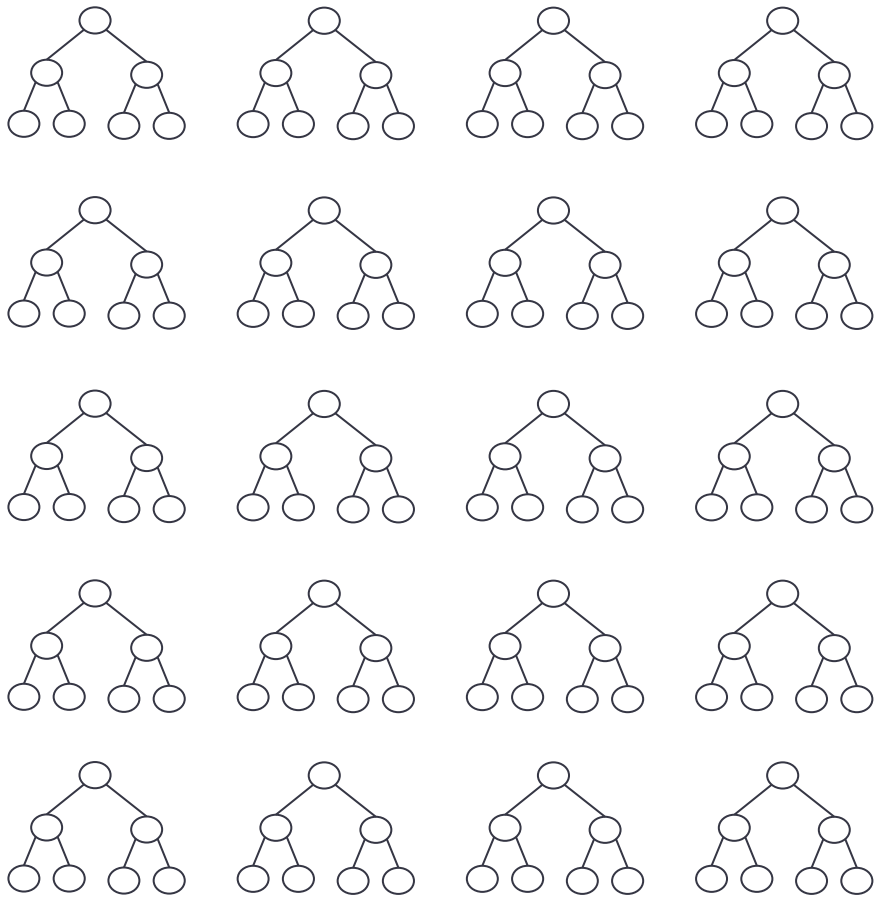


Massachusetts Institute of Technology

CSAIL

DP for DEC-POMDPs continued



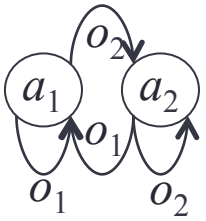


Infinite horizon version (Bernstein et al., 09)

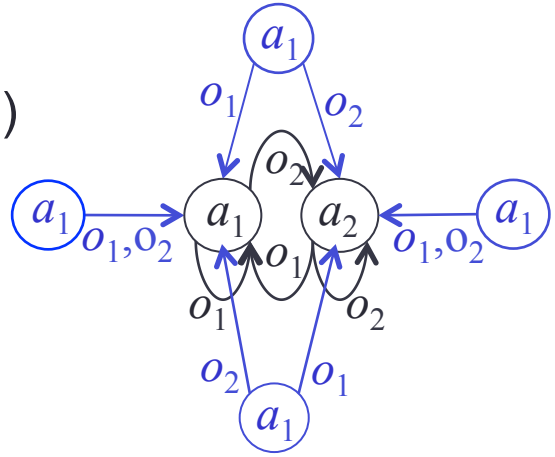
- Remember we need to define a controller for each agent
- How many nodes do you need and what should the parameter values be for an optimal *infinite-horizon* policy?
- This may be infinite!
- First ε-optimal algorithm for infinite-horizon Dec-POMDPs: Policy Iteration

Optimal DP: Policy Iteration

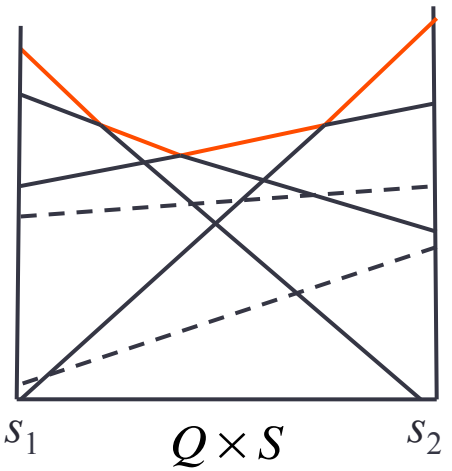
- Start with a given controller
- Exhaustive backup (for all agents): generate all next step policies
- Evaluate: determine value of starting at each node at each state and for each policy for the other agents
- Prune: remove those that always have lower value (merge as needed)
- Continue with backups and pruning until error is below ε



= Initial controller for agent 1



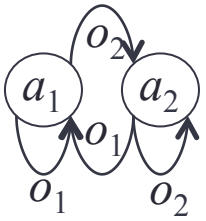
(backup for action 1)



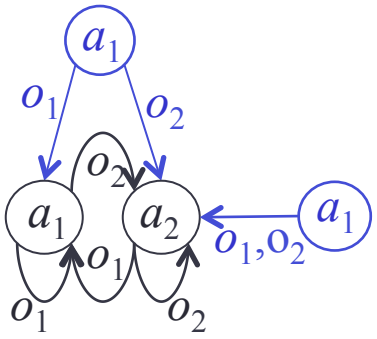
Optimal DP: Policy Iteration

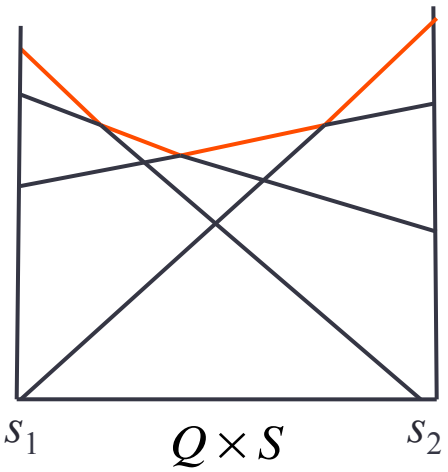
- Start with a given controller
- Exhaustive backup (for all agents): generate all next step policies
- Evaluate: determine value of starting at each node at each state and for each policy for the other agents
- Prune: remove those that always have lower value (merge as needed)
- Continue with backups and pruning until error is below ε

Key: Prune over not just states, but possible policies of the other agents!



= Initial controller for agent 1





(backup for action 1)

Massachusetts Institute of Technology

CSAIL

DP for for Dec-POMDPs

- What happens in infinite or large horizons?
  - Number of trees is doubly exponential in the horizon
  - Doesn't consider start state
  - Full backup wasteful (many trees pruned)
- Need more efficient backups
- Or approximate approaches to increase scalability

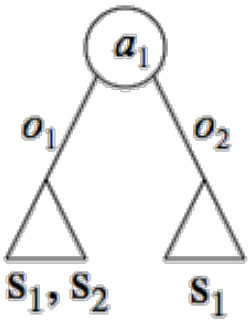
Incremental policy generation (Amato et al., 09)

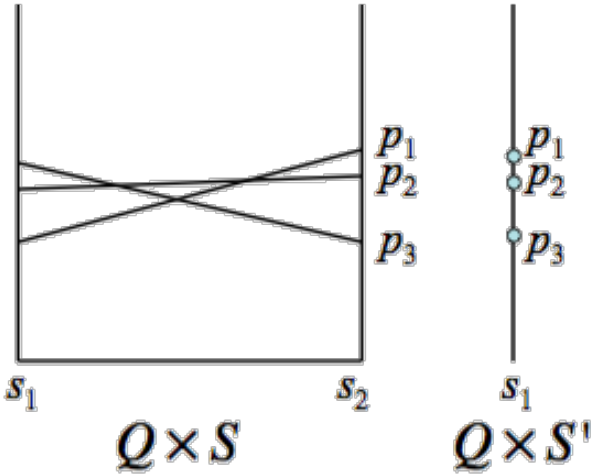
- Optimal dynamic programming for Dec-POMDPs requires a large amount of time and space
- In POMDPs, methods have been developed to make optimal DP more efficient (e.g., incremental pruning)
- These cannot be extended to Dec-POMDPs (due to the lack of a shared viewpoint by the agents)
- Makes the optimal approaches for both finite and infinite-horizon more efficient

Incremental policy generation (cont.)

- Can avoid exhaustively generating policies (backups)
- Can limit the number of states considered based on action and observation (see a wall, other agent, etc.)
- This allows policies for an agent to be built up incrementally
- Add only subtrees (or subcontrollers) that are not dominated

Key: rune only over reachable subspaces

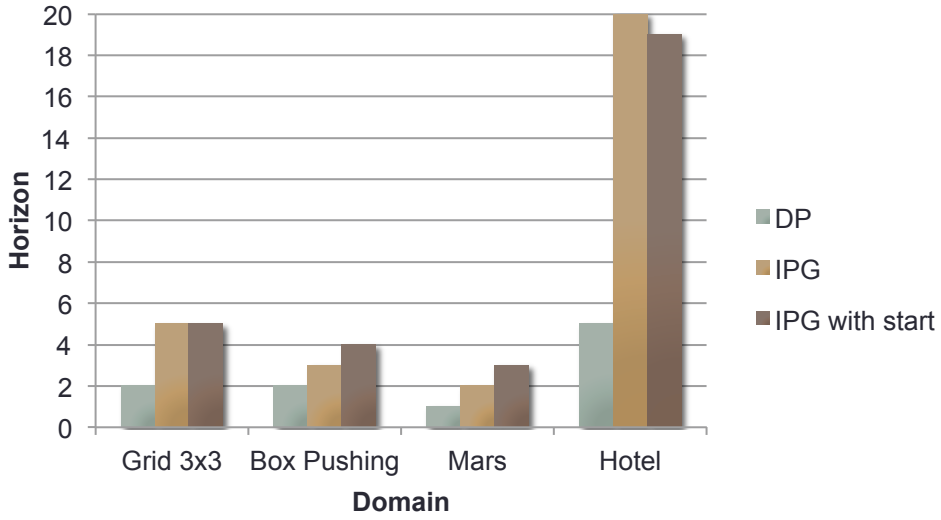




State-of-the-art in bottom up and infinite-horizon

Benefits of IPG and results

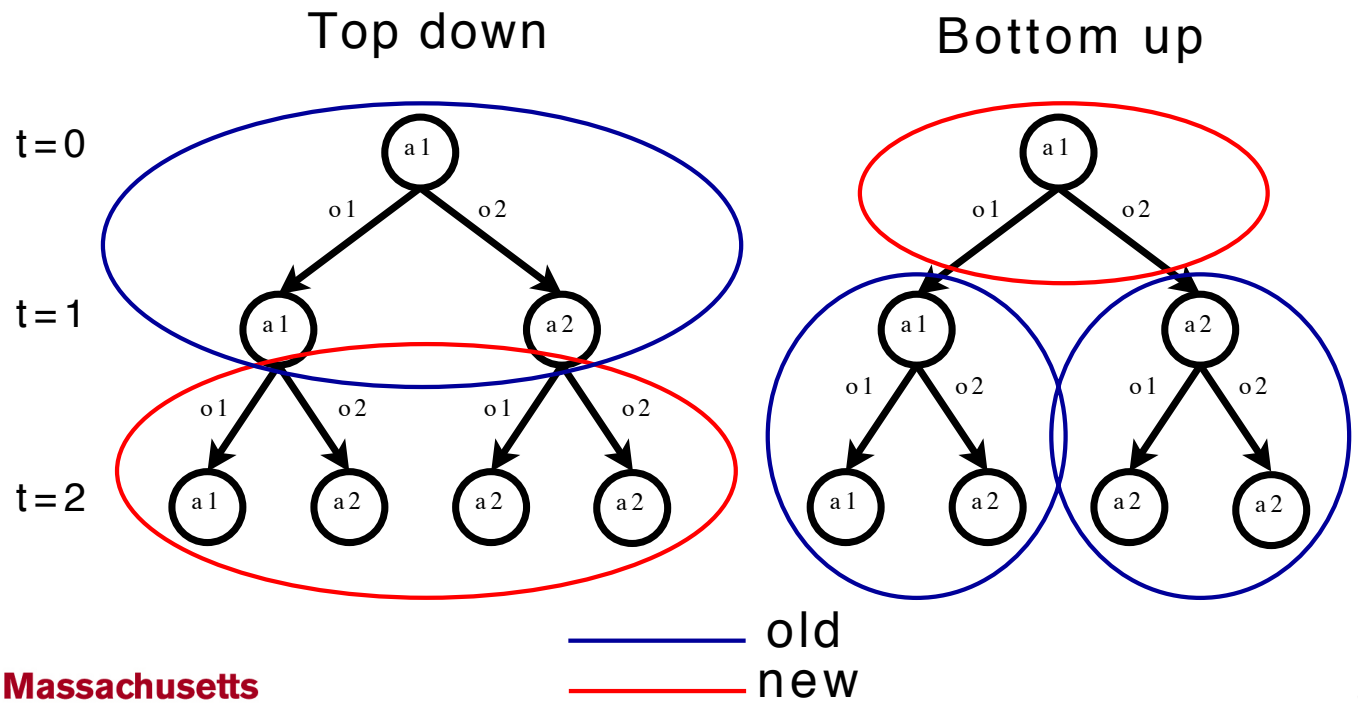
- Solve larger problems optimally
- Can make use of start state information as well
- Can be used in other dynamic programming algorithms
  - Optimal: Finite-, infinite- and indefinite horizon
  - Approximate: PBDP, MBDP, IMBDP, MBDP-OC and PBIP



Increases horizon in optimal DP (finite or infinite-hor)

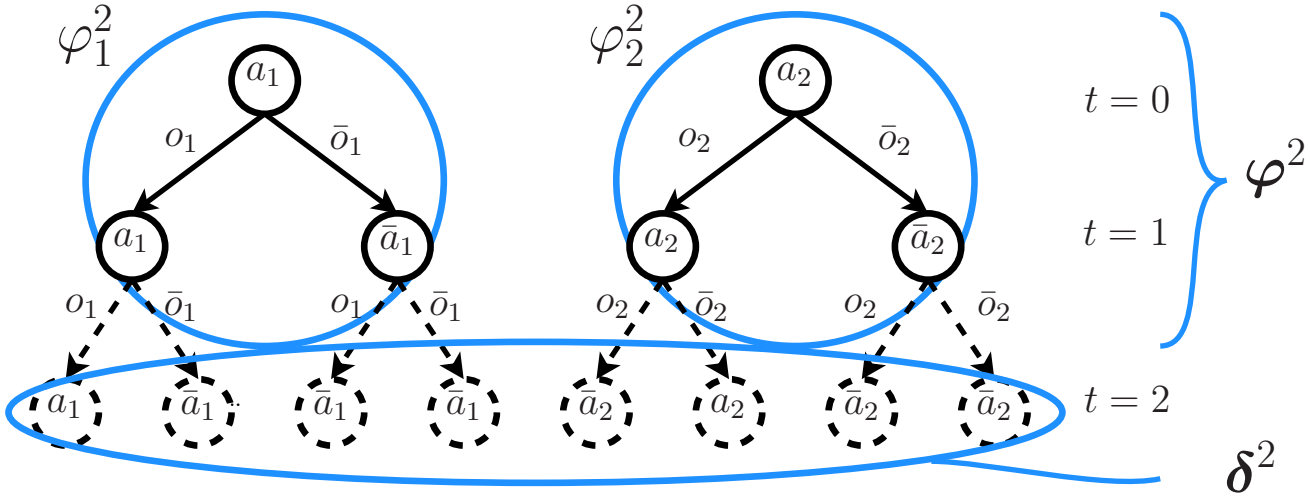
Top down approaches

- Perform a search starting from a known initial (belief) state
- Continue until the desired horizon is reached



Multiagent A* (Szer et al., 05)

- Can build up policies for agents from the first step
- Use heuristic search over *joint* policies
  - Actions: a and $\bar{a}$, observations: o and $\bar{o}$
  - History: φ and joint policies for a single step: δ
  - Heuristic value for remaining steps (e.g., MDP or POMDP)



Massachusetts Institute of Technology

CSAIL

Multiagent A* continued

- Requires an admissible heuristic function
- A*-like search over partially specified joint policies:

$$\varphi^t = (\delta^0, \delta^1, \ldots, \delta^{t-1}),$$

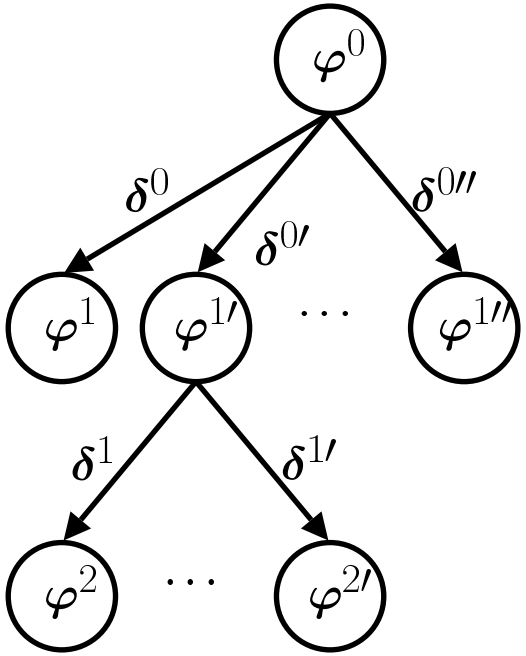
$$\delta^t = (\delta_0^t, \ldots, \delta_n^t) \qquad \delta_i^t : \vec{\mathcal{O}}_i^t \rightarrow A_i$$

- Heuristic value: $\underbrace{\widehat{V}(\varphi^t)}_{F} = \underbrace{V^{0\ldots t-1}(\varphi^t)}_{G} + \underbrace{\widehat{V}^{t\ldots h-1}}_{H}$

- If $\widehat{V}^{t\ldots h-1}$ is admissible (overestimation), so is $\widehat{V}(\varphi^t)$

Multiagent A* continued

- Expand children by growing policy to the next horizon
- Choose nodes with the highest *F* value
- Continue until node with highest *F* is a full horizon policy



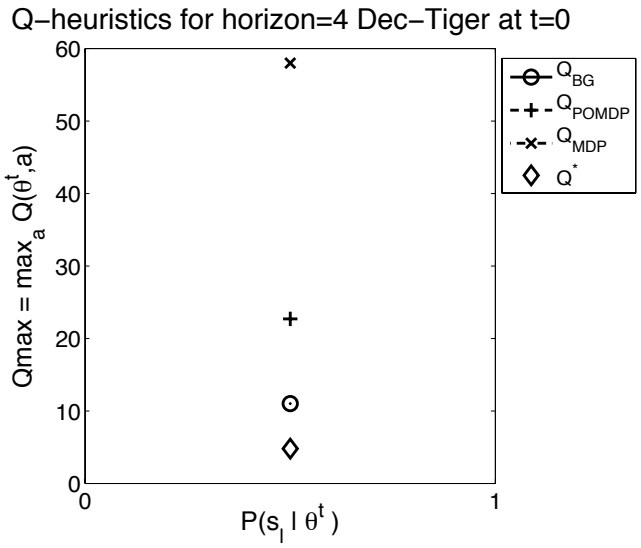
$$\varphi^t = (\delta^0, \delta^1, \ldots, \delta^{t-1}),$$

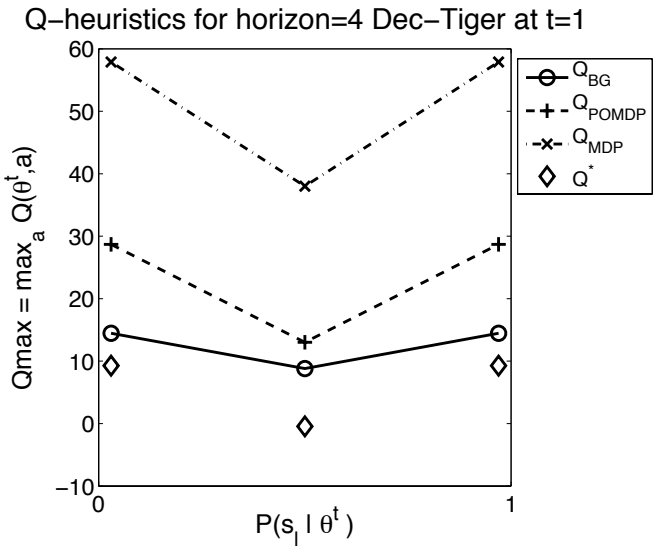
$$\delta^t = (\delta_0^t, \ldots, \delta_n^t) \qquad \delta_i^t : \vec{\mathcal{O}}_i^t \to A_i$$

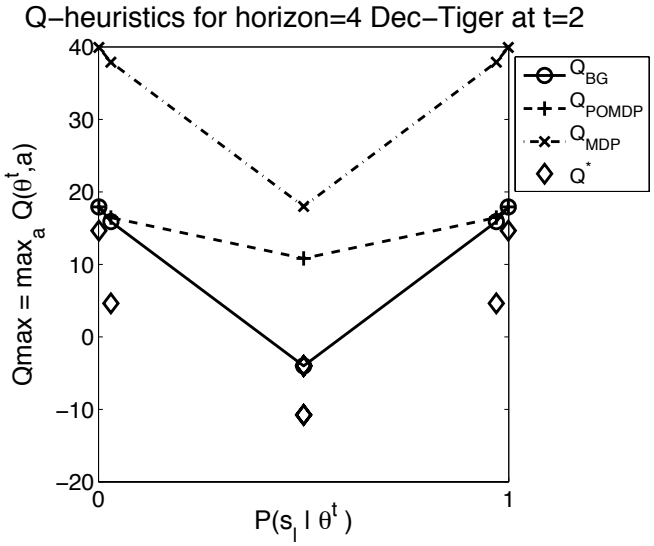
Heuristic functions

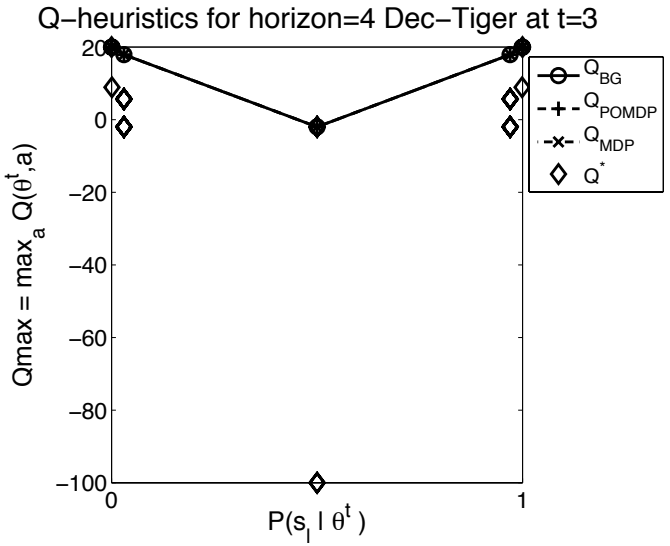
- Defined by solving simplified problem settings
- Q_{MDP}: assume the underlying state is fully observable by the agents from that step on
  - Cheap to compute (solve an MDP)
  - Often loose (strong assumptions: centralized and fully observable)
- Q_{POMDP}: assume the observations of all agents are shared from that step on
  - More difficult to solve (exponential in *h* to solve POMDP)
- Q_{BG}: assume the observations of all agents are shared *on the next step* on
  - Still harder to solve
- Hierarchy of upper bounds $Q^* \leq Q_{BG} \leq Q_{PODP} \leq Q_{MDP}$

Heuristic functions: example tiger problem



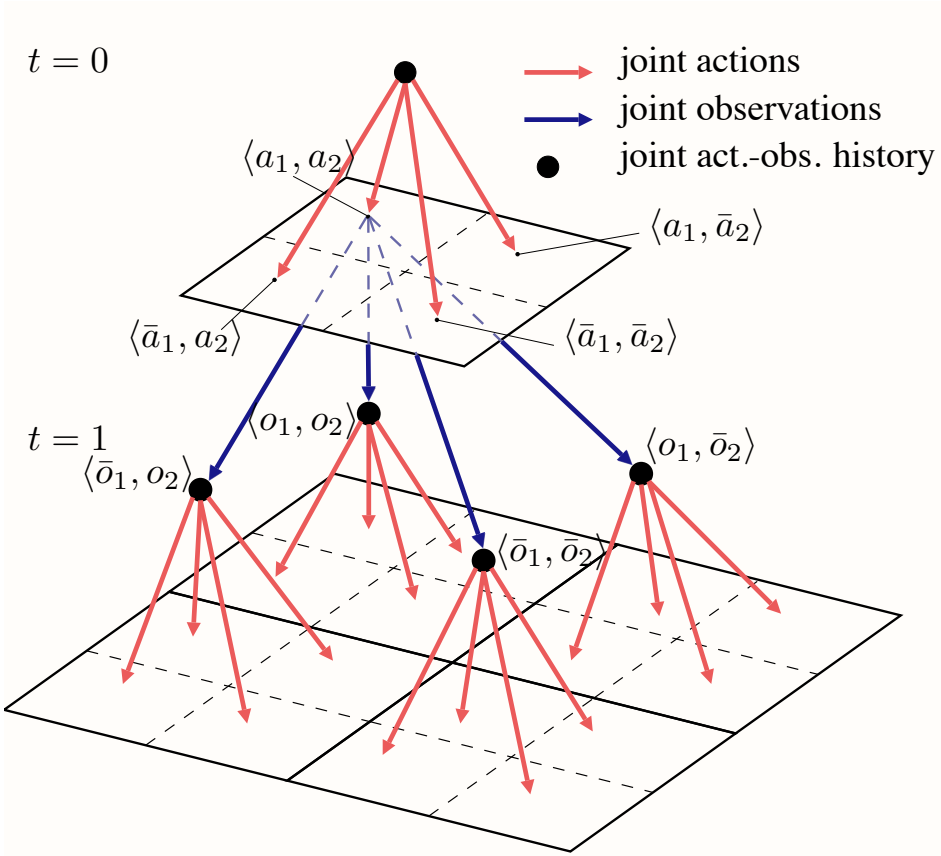






Generalized MAA* (Oliehoek et al., 08)

- Represent Dec-POMDPs as a series of Bayesian games
- Also allowed different heuristic functions and solvers to be used



Massachusetts Institute of Technology

CSAIL

Dec-POMDPs as a series of Bayesian games

$\vec{\theta}_1^{t=0}$	$\vec{\theta}_2^{t=0}$	$()$	
		a_2	$\bar{a}_2$
$()$	a_1	$+2.75$	-4.1
	$\bar{a}_1$	-0.9	$+0.3$

$\vec{\theta}_1^{t=1}$	$\vec{\theta}_2^{t=1}$	(a_2, o_2)		$(a_2, \bar{o}_2)$		...
		a_2	$\bar{a}_2$	a_2	$\bar{a}_2$	
(a_1, o_1)	a_1	-0.3	$+0.6$	-0.6	$+4.0$	...
	$\bar{a}_1$	-0.6	$+2.0$	-1.3	$+3.6$	...
$(a_1, \bar{o}_1)$	a_1	$+3.1$	$+4.4$	-1.9	$+1.0$	...
	$\bar{a}_1$	$+1.1$	-2.9	$+2.0$	-0.4	
$(\bar{a}_1, o_1)$	a_1	-0.4	-0.9	-0.5	-1.0	...
	$\bar{a}_1$	-0.9	-4.5	-1.0	$+3.5$	...
$(\bar{a}_1, \bar{o}_1)$	...	...	...	...	...	...

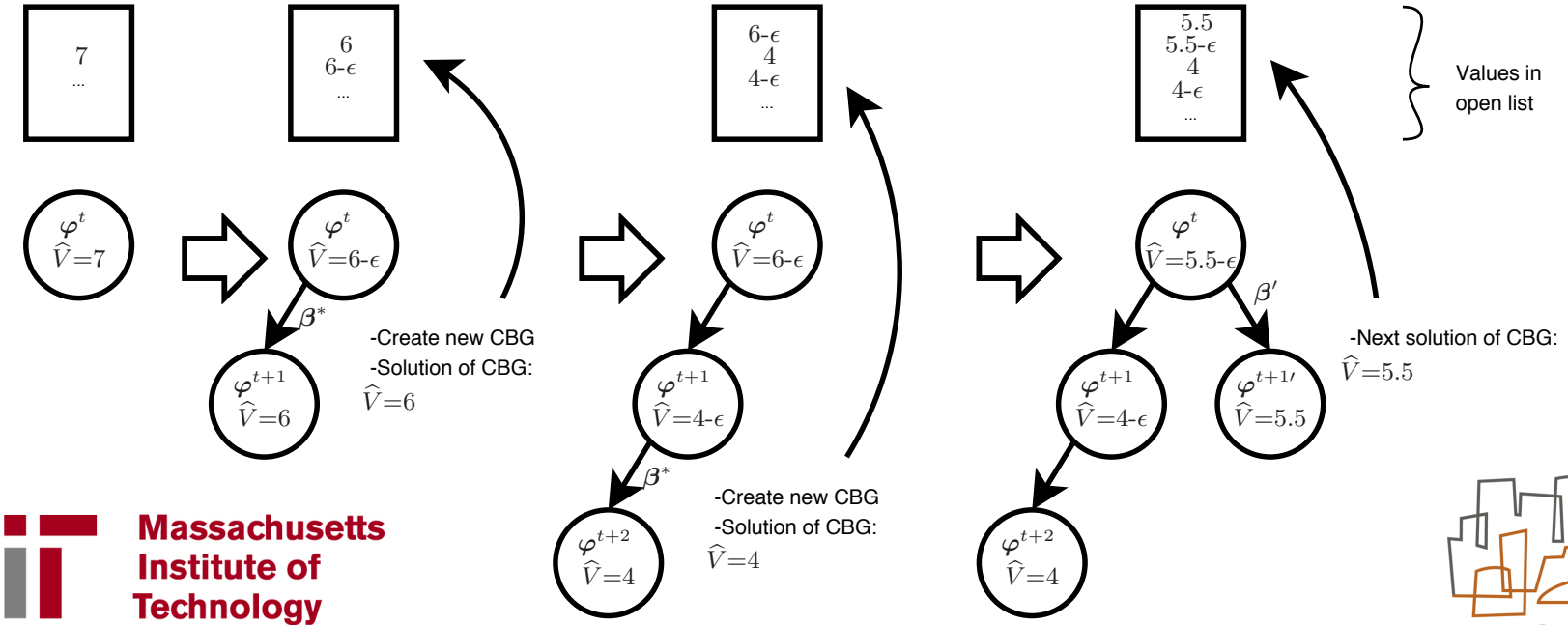
CSAIL

Lossless clustering of histories (Oliehoek et al, 09)

- Idea: if two individual histories induce the same distribution over states and over other agents' histories, they are equivalent and can be clustered
- Lossless clustering, independent of heuristic, but problem dependent
- Clustering is bootstrapped: algorithms only deal with clustered Bayesian games
- Large increase in scalability of optimal solvers

Incremental expansion (Spaan et al., 11)

- Number of children expanded is doubly exponential in *t*
- Many of these not useful for optimal policy
- Use incremental expansion
  - Find next step policies as a cooperative Bayesian game (CBG)
  - Keep pointer to unexpanded nodes in open list

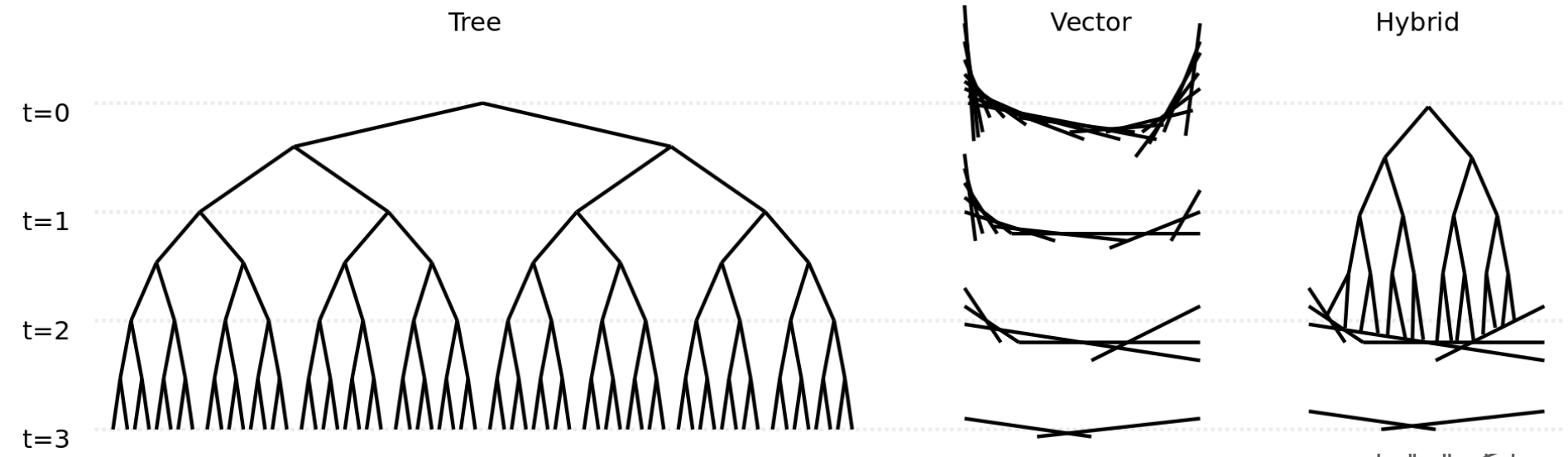


Massachusetts Institute of Technology

CSAIL

Hybrid heuristic

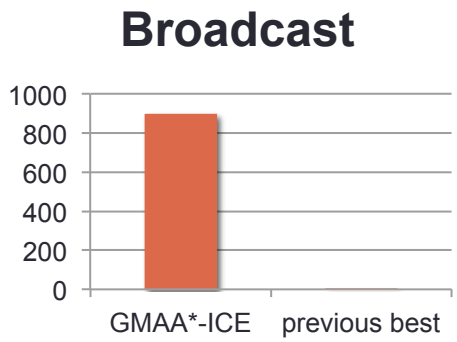
- Two standard representations for heuristics
  - Tree: values for all joint action-observation histories
  - Vector: a potentially exponential number of vectors
- Key insight: exponential growth of these representations is in opposite directions

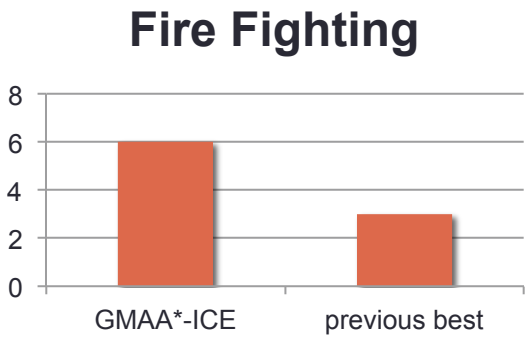


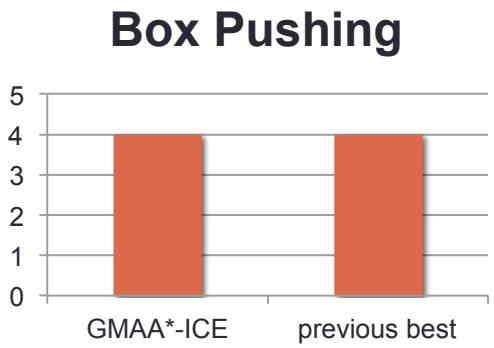
State-of-the-art in top down and finite-horizon

Optimal results

- Problem size (all 2 agent)
  - Broadcast Channel S=4 A=2 Ω=2
  - Box Pushing S=100 A=4 Ω=5
  - Fire Fighting S=432 A=3 Ω=2
- Performance a combination of better search and better heuristics







Massachusetts Institute of Technology

CSAIL

Approximate methods

- Optimal methods are intractable for many problems
- Want to produce the best solution possible with given resources
- Quality bounds are usually not possible, but these approaches often perform well

Joint equilibrium search for policies (JESP)

(Nair et al., 03)

- Instead of exhaustive search, find best response

- Algorithm:

    Start with (full) policy for each agent

    while *not converged* do

        for *i*=1 to *n*

            Fix other agent policies

            Find a best response policy for agent *i*

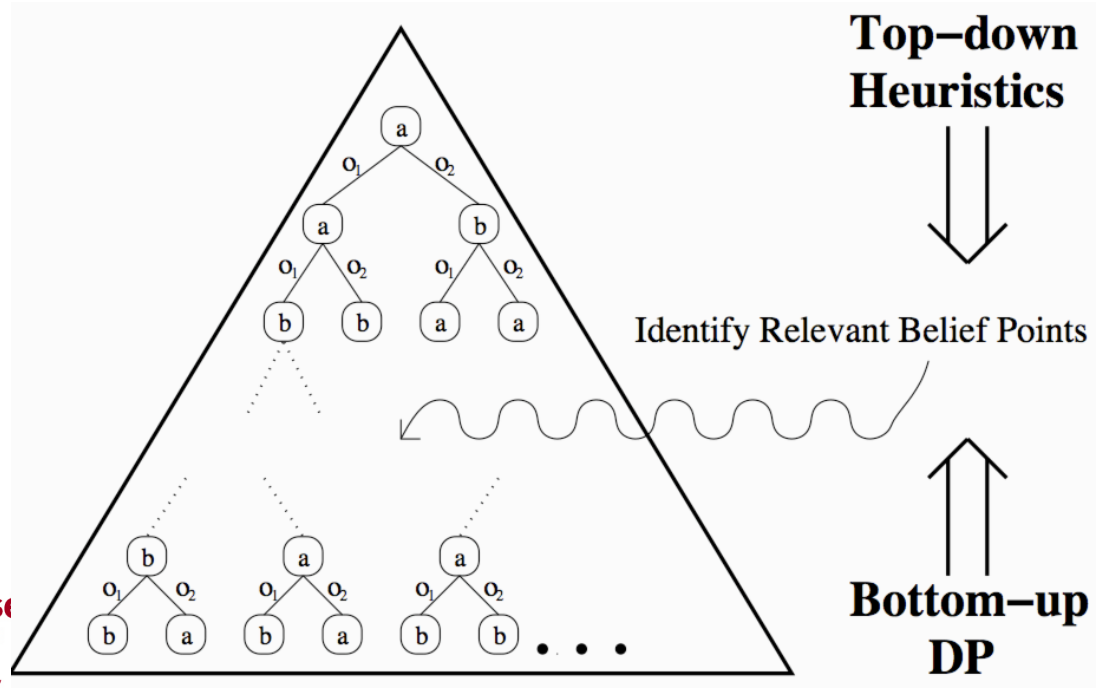
JESP summary

- Finds a *locally optimal* set of policies
- Worst case complexity is the same as exhaustive search, but in practice is much faster
- Can also incorporate dynamic programming to speed up finding best responses
  - Fix policies of other agents
  - Create a (augmented) POMDP using the fixed policies of others
  - Generate reachable belief states from initial state b_0
  - Build up policies from last step to first
  - At each step, choose subtrees that maximize value at reachable belief states

Memory bounded dynamic programming (MBDP)

(Seuken and Zilberstein., 07)

- Do not keep all policies at each step of dynamic programming
- Keep a fixed number for each agent: *maxTrees*
- Select these by using heuristic solutions from initial state
- Combines top down and bottom up approaches



MBDP algorithm

start with a one-step policy for each agent

for *t=h* to *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 b_0

MBDP summary

- Linear complexity in problem horizon
- Exponential in the number of observations
- Performs well in practice (often with very small *maxTrees*)
- Can be difficult to choose correct *maxTrees*

Extensions to MBDP

- IMBDP: Limit the number of observations used based on
- probability at each belief (Seuken and Zilberstein 07)
- MBDP-OC: compress observations based on the value produced (Carlin and Zilberstein 08)
- PBIP: heuristic search to find best trees rather than exhaustive (Dibangoye et al., 09)
- Current state-of-the-art
  - PBIP-IPG: extends PBIP by limiting the possible states (Amato et al., 09-AAMAS)
  - CBPB: uses constraint satisfaction solver for subtree selection (Kumar and Zilberstein, 10)
  - PBPG: approximate, linear programming method for subtree selection (Wu et al, 10) – solves a problem with 3843 states and 11 obs to hor 20

Other finite-horizon approaches

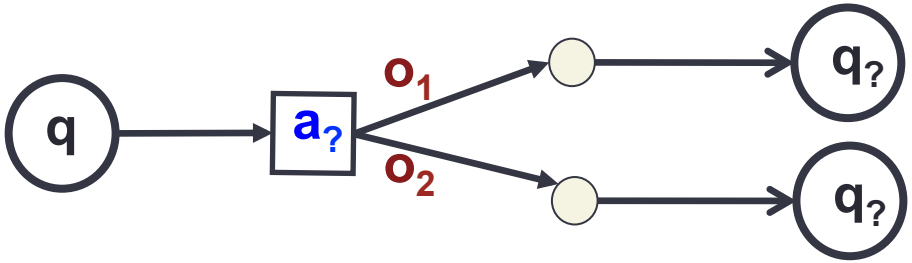
- Mixed integer linear programming (MILP) (Aras et al., 07)
  - Represent each agent's policy in sequence form (instead of as a tree)
  - Solve as a combinatorial optimization problem (MILP)
- Sampling methods
  - Direct Cross-Entropy policy search (DICE) (Oliehoek et al., 08)
    - Randomized algorithm using combinatorial optimization
    - Applies Cross-Entropy method to Dec-POMDPs
    - Scales well wrt number of agents
  - Goal-directed sampling (Amato and Zilberstein, 09)
    - Discussed later

Approximate infinite-horizon approaches

- 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:
  - Best-First Search (BFS)
  - Bounded Policy Iteration for Dec-POMDPs (Dec-BPI)
  - Nonlinear Programming (NLP)

Memory-bounded solutions

- Optimal approaches may be intractable
- Can use fixed-size finite-state controllers as policies for Dec-POMDPs
- How do we set the parameters of these controllers to maximize their value?
  - Deterministic controllers - discrete methods such as branch and bound and best-first search
  - Stochastic controllers - continuous optimization



(deterministically) choosing an action and transitioning to the next node

Best-first search (Szer and Charpillet 05)

- Search through space of deterministic action selection and node transition parameters
- Produces optimal fixed-size deterministic controllers
- High search time limits this to very small controllers (< 3 nodes)

Bounded policy iteration (BPI) (Bernstein et al., 05)

- Improve the controller over a series of steps until value converges
- Alternate between improvement and evaluation
- Improvement
  - Use a linear program to determine if a node's parameters can be changed, while fixing the rest of the controller and other agent policies
  - Improved nodes must have better value for all states and nodes of the other agents (multiagent belief space)
- Evaluation: Update the value of all nodes in the agent's controller
- Can solve much larger controller than BFS, but value is low due to lack of start state info and LP

Nonlinear programming approach (Amato et al., 07, 09b)

- Use a nonlinear program (NLP) to represent an optimal fixed-size set of controllers for Dec-POMDPs
- Consider node value as well as action and transition parameters as variables
- Maximize the value using a known start state
- Constraints maintain valid values and probabilities

NLP formulation

Variables: $x(q_i, a_i) = P(a_i \mid q_i), y(q_i, a_i, o_i, q_i') = P(q_i' \mid q_i, a_i, o_i), z(\vec{q}, s) = V(\vec{q}, s)$

Objective: Maximize $\sum_s b_0(s) z(\vec{q}_0, s)$

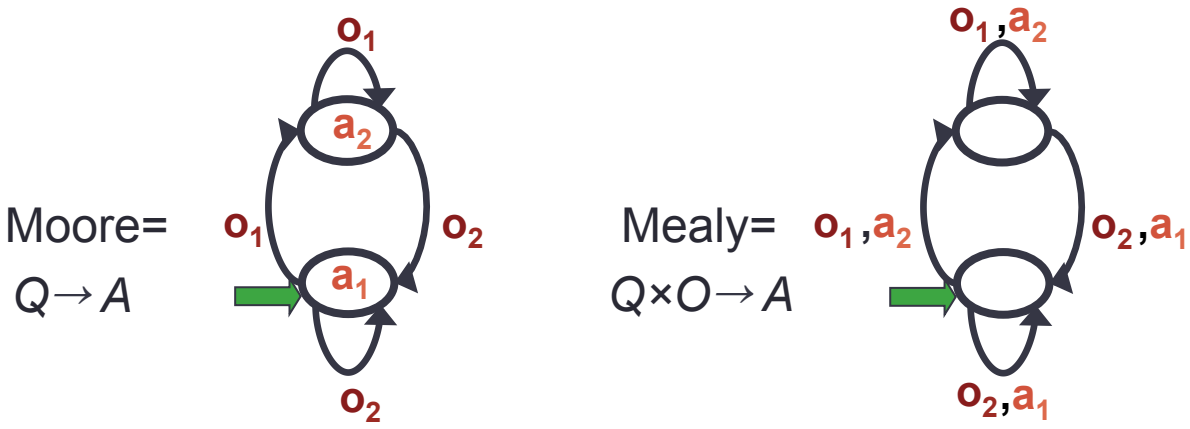
Value Constraints: $\forall s \in S, \vec{q} \in Q$

$$z(\vec{q}, s) = \sum_a \left(\prod_i x(q_i, a_i) \left[R(s, \vec{a}) + \gamma \sum_{s'} P(s' \mid s, \vec{a}) \sum_o O(\vec{o} \mid s', \vec{a}) \sum_{q'} \prod_i y(q_i', a_i, q_i, o_i) z(\vec{q}', s') \right] \right)$$

Probability constraints ensure all probabilities must sum to 1 and be greater than 0

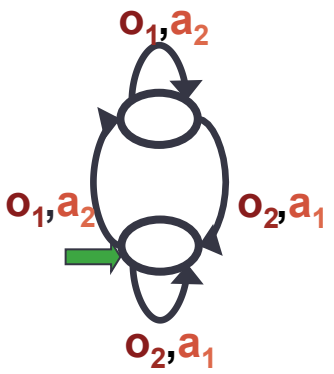
Mealy controllers (Amato et al, 10)

- Controllers currently used are Moore controllers
- Mealy controllers are more powerful than Moore controllers (can represent higher quality solutions with the same number of nodes)
- Key difference: action depends on node and observation



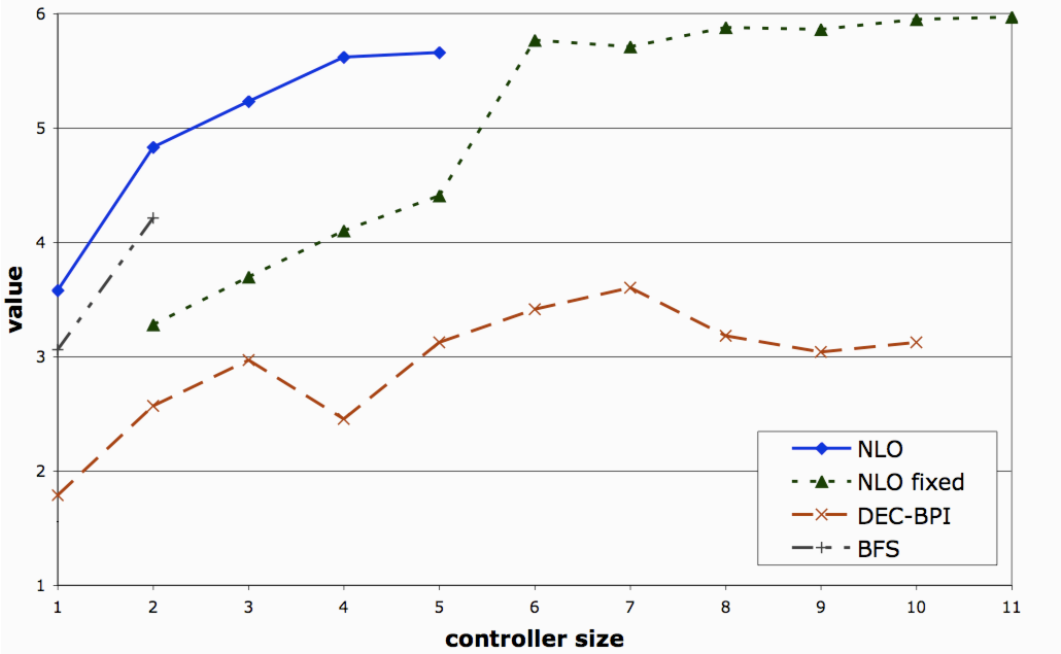
Mealy controllers continued

- More powerful
- Provides extra structure that algorithms can use
  - Can automatically simplify representation based on informative observations
  - Can be done in controller or solution method
- Can be used in place of Moore controllers in all controller-based algorithms for POMDPs and DEC-POMDPs (not just NLP)
  - Optimal infinite-horizon DP
  - Approximate algorithms



Some infinite-horizon results

- Optimal algorithm can only solve very small problems
- Approximate algorithms are more scalable



- GridSmall: 16 states, 4 actions, 2 obs
- Policy Iteration: 3.7 with 80 nodes in 821s before out of memory

Indefinite-horizon

- Unclear how many steps are needed until termination
- Many natural problems terminate after a goal is reached
  - Meeting or catching a target
  - Cooperatively completing a task

Indefinite-horizon Dec-POMDPs (Amato et al, 09a)

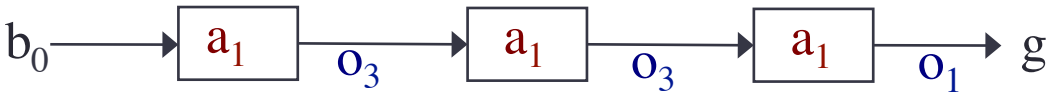
- Extends indefinite-horizon POMDPs Patek 01 and Hansen 07
- Our assumptions
  - Each agent possesses a set of terminal actions
  - Negative rewards for non-terminal actions
- Can capture uncertainty about reaching goal
- Many problems can be modeled this way

An optimal solution to this problem can be found using dynamic programming

Goal-directed Dec-POMDPs

- Relax assumptions, but still have goal
- Problem terminates when
  - A single agent or set of agents reach local or global goal states
  - Any combination of actions and observations is taken or seen
- More problems fall into this class (can terminate without agent knowledge)
- Solve by *sampling* trajectories
  - Produce only action and observation sequences that lead to goal
  - This reduces the number of policies to consider

Can bound the number of samples required to approach optimality



State-of-the-art in infinite-horizon approximate

Infinite and indefinite-horizon results

- Standard infinite-horizon benchmarks
- Approximate solutions
- Can provide a very high-quality solution very quickly in each problem

Algorithm	Value	Size	Time
Two Agent Tiger: $\|S\| = 2, \|A_i\| = 3, \|O_i\| = 2$			
HPI w/ NLP	6.80	6	119
Goal-directed	5.04	12	75
Moore	-1.09	19	6,173
Meeting in a Grid: $\|S\| = 16, \|A_i\| = 5, \|O_i\| = 2$			
Mealy	6.13	5	116
HPI w/ NLP	6.04	7	16,763
Moore	5.66	5	117
Goal-directed	5.64	4	4
Box Pushing: $\|S\| = 100, \|A_i\| = 4, \|O_i\| = 5$			
Goal-directed	149.85	5	199
Mealy	143.14	4	774
HPI w/ NLP	95.63	10	6,545
Moore	50.64	4	5,176
Mars Rover: $\|S\| = 256, \|A_i\| = 6, \|O_i\| = 8$			
Goal-directed	21.48	6	956
Mealy	19.67	3	396
HPI w/ NLP	9.29	4	111
Moore	8.16	2	43

Scalability to larger problems

- Dec-MDP
  - Collective observations fully determine state
- Independent transitions and observations
  - $P(s' | s, a) = P(s' | s, a_1)\, P(s' | s, a_2)$
  - $O(o | s', a) = P(o_1 | s', a_1)\, P(o_2 | s', a_2)$
  - Rewards still dependent
- Complexity drops to NP (from NEXP)
- Policy: nonstationary map from local states to actions $s_i^t \rightarrow a$
- Many realistic problems have this type of independence

Sample-based heuristic search under submission

- Observation: can share policies before execution
  - With Dec-MDP assumptions, can now estimate system state
  - Don't need history, so can use this estimate and single local obs
- Use branch and bound to search policy space
  - Start from known initial state
- Incorporate constraint optimization to more efficiently expand children

State-of-the-art in IT-IO Dec-MDPs

Dec-MDP results

- 2 agent problems

T	$v_0(\eta_0)$	ICE	IPG	BLP	*COP*
Meeting on a 8x8 Grid ($\|Z\| = 4096, \|A\| = 25$)					
5	0.0	12.55	-	-	5.051
6	**0.0**	-	-	-	**6.20**
7	**0.71**	-	-	-	**13.16**
8	**1.67**	-	-	-	**13.76**
9	**2.68**	-	-	-	**16.94**
10	**3.68**	-	-	-	**18.66**
20	**13.68**	-	-	-	**38.37**
30	**23.68**	-	-	-	**52.39**
40	**33.68**	-	-	-	**59.70**
50	**43.68**	-	-	-	**74.28**
100	**93.68**	-	-	-	**214.73**
Navigation (MIT) ($\|Z\| = 7225, \|A\| = 16$)					
10	0.0	85.85	-	-	47.322
20	**0.0**	-	-	-	**321.26**
40	**34.97**	-	-	-	**400.48**
50	**54.92**	-	-	-	**1061.94**
100	**154.93**	-	-	-	**1236.11**

T	$v_0(\eta_0)$	ICE	IPG	BLP	*COP*
Navigation (ISR) ($\|Z\| = 8100, \|A\| = 16$)					
2	0.0	0.71	-	3225.8	2.39
3	0.0	6.50	-	-	3.19
4	**0.0**	-	-	-	**4.31**
5	**0.38**	-	-	-	**13.43**
10	**6.47**	-	-	-	**54.16**
50	**83.02**	-	-	-	**194.66**
100	**182.54**	-	-	-	**1294.28**
Navigation (PENTAGON) ($\|Z\| = 9801, \|A\| = 16$)					
2	0.0	0.99	-	4915.1	1.31
3	0.0	6.01	-	-	5.11
4	**0.0**	-	-	-	**8.75**
5	**0.38**	-	-	-	**13.81**
10	**4.82**	-	-	-	**62.89**
20	**19.73**	-	-	-	**129.48**
30	**39.18**	-	-	-	**209.11**
40	**57.75**	-	-	-	**276.20**
50	**76.39**	-	-	-	**1033.70**

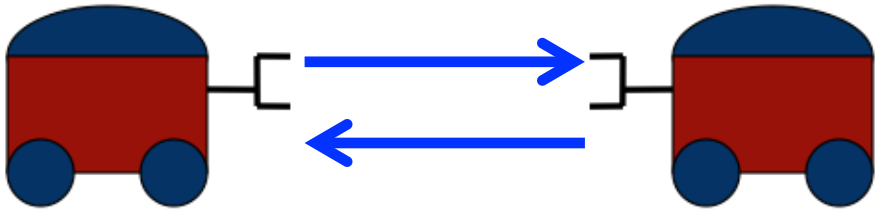
- Scalable to 6 agents for small problems and horizons
- Note: ND-POMDP methods can scale to up to 15 agents by making reward decomposability assumptions

Massachusetts Institute of Technology

CSAIL

Communication

- Communication can be implicitly represented in Dec-POMDP model
- Free and instantaneous communication is equivalent to centralization
- Otherwise, need to reason about what and when to communicate



Dec-POMDP-COM

- A DEC-POMDP-COM can be defined with the tuple:

M = <I, S, {A_i}, P, R, {Ω_i}, O, Σ,C_Σ>

  - The first parameters are the same as a Dec-POMDP
  - Σ, the alphabet of atomic communication messages for each agent (including a null message)
  - C_Σ, the cost of transmitting an atomic message: $C_\Sigma : \Sigma \rightarrow \Re$
  - R, the reward model integrates this cost for the set of agents sending message σ: $R(s,\vec{a},\vec{\sigma})$
- Explicitly models communication
- Same complexity as Dec-POMDP

Algorithms using communication

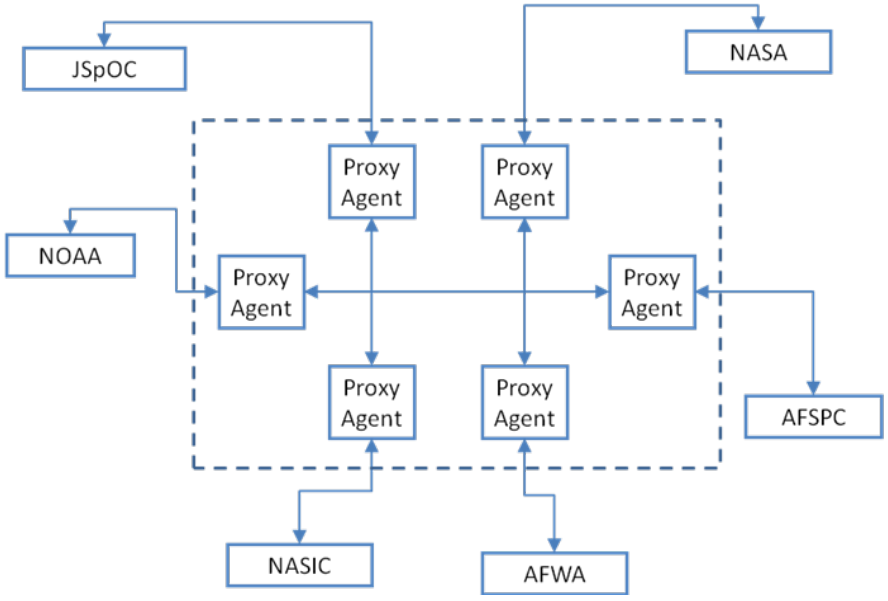
- Analysis of possible communication models and complexity (Pynadath et al., 02)
- Myopic communication in transition independent Dec-MDPs (Becker et al., 09)
- Reasoning about run-time communication decisions (Nair et al., 04; Roth et al., 05)
- Stochastically delayed communication (Spaan et al., 08)

Summary

- Optimal algorithms for Dec-POMDPs give performance guarantees, but are often intractable
  - Top down and bottom up methods provide similar performance
- Approximate Dec-POMDP algorithms are much more scalable, but (often) lack quality bounds
  - Bounding memory and sampling are dominant approaches
- Using subclasses can significantly improve solution scalability (if assumptions hold)
- Communication can be helpful, but difficult to decide when and how to communicate

Application: Personal assistant agents (Amato et al.,11)

- People connected to many others and sources of info
- Use software personal assistant agents to provide support (Dec-MDP, Shared-MDP)
  - Agents collaborate on your behalf to find resources, teams, etc.
  - Goal: work more efficiently with others and discover helpful info

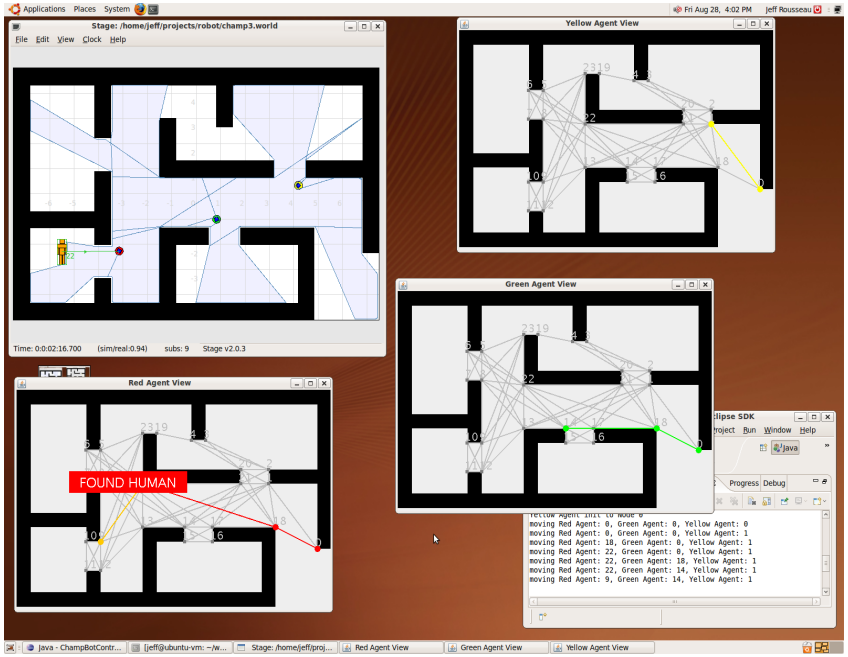


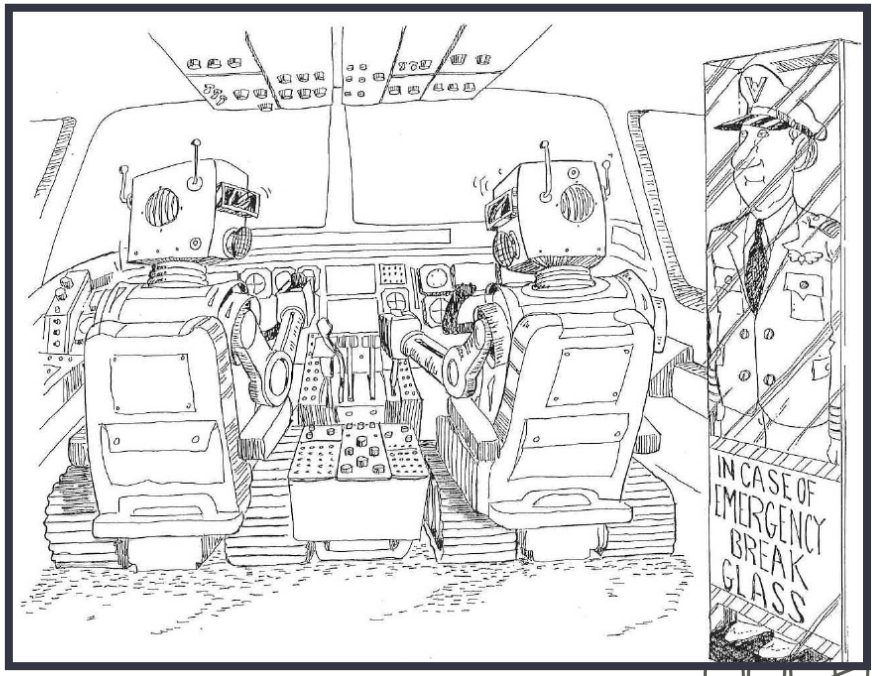
Application: Unmanned system control

- Planning for autonomous vehicles in a simulation
  - Single or centralized team: factored POMDP
  - Decentralized team: Dec-POMDP
  - Team considering adaptive enemy: I-POMDP

Application: Mixed-initiative robotics

- Humans and robots collaborating for search and pursuit
- Determine what tasks robots should do (MMDP, Dec-POMDP) and what tasks humans should do
- Adjustable autonomy based on preferences and situation





Massachusetts Institute of Technology

APTIMA HUMAN-CENTERED ENGINEERING

CSAIL

Conclusion

- What problems Dec-POMDPs are good for
  - Sequential (not “one shot” or greedy)
  - Cooperative (not single agent or competitive)
  - Decentralized (not centralized execution or free, instantaneous communication)
  - Decision-theoretic (probabilities and values)

Resources

- My Dec-POMDP webpage
  - Papers, talks, domains, code, results
  - http://rbr.cs.umass.edu/~camato/decpomdp/
- Matthijs Spaan's Dec-POMDP page
  - Domains, code, results
  - http://users.isr.ist.utl.pt/~mtjspaan/decpomdp/index_en.html
- USC's Distributed POMDP page
  - Papers, some code and datasets
  - http://teamcore.usc.edu/projects/dpomdp/
- Our full tutorial "Decision Making in Multiagent Systems"
  - http://users.isr.ist.utl.pt/~mtjspaan/tutorialDMMS/

References

- **Scaling Up Optimal Heuristic Search in Dec-POMDPs via Incremental Expansion.** Frans A. Oliehoek, Matthijs T. J. Spaan and Christopher Amato. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (IJCAI-11), Barcelona, Spain, July, 2011
- **Towards Realistic Decentralized Modelling for Use in a Real-World Personal Assistant Agent Scenario.** Christopher Amato, Nathan Schurr and Paul Picciano. In *Proceedings of the Workshop on Optimisation in Multi-Agent Systems (OptMAS) at the Tenth International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS-11), Taipei, Taiwan, May 2011.
- **Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs.** Christopher Amato, Blai Bonet and Shlomo Zilberstein. *Proceedings of the Twenty-Fourth National Conference on Artificial Intellgence* (AAAI-10), Atlanta, GA, July, 2010.
- **Point-based Backup for Decentralized POMDPs: Complexity and New Algorithms.** Akshat Kumar and Shlomo Zilberstein. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1315-1322, 2010.
- **Point-Based Policy Generation for Decentralized POMDPs**, Feng Wu, Shlomo Zilberstein, and Xiaoping Chen,
Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-10), Page 1307- 1314, Toronto, Canada, May 2010.
- **Optimizing Fixed-size Stochastic Controllers for POMDPs and Decentralized POMDPs.** Christopher Amato, Daniel S. Bernstein and Shlomo Zilberstein. *Journal of Autonomous Agents and Multi-Agent Systems* (JAAMAS) 2009.
- **Incremental Policy Generation for Finite-Horizon DEC-POMDPs.** Christopher Amato, Jilles Steeve Dibangoye and Shlomo Zilberstein. *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling* (ICAPS-09), Thessaloniki, Greece, September, 2009.

References continued

- **Event-detecting Multi-agent MDPs: Complexity and Constant-Factor Approximation.** Akshat Kumar and Shlomo Zilberstein. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 201-207, 2009.
- **Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs**. Jilles S. Dibangoye, Abdel-Illah Mouaddib, and Brahim Chaib-draa. In *Proc. of the Joint International Conference on Autonomous Agents and Multi-Agent Systems* , 2009.
- **Constraint-Based Dynamic Programming for Decentralized POMDPs with Structured Interactions.** Akshat Kumar and Shlomo Zilberstein In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 561-568, 2009.
- **Achieving Goals in Decentralized POMDPs.** Christopher Amato and Shlomo Zilberstein. *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS-09), Budapest, Hungary, May, 2009.
- **Policy Iteration for Decentralized Control of Markov Decision Processes.** Daniel S. Bernstein, Christopher Amato, Eric A. Hansen and Shlomo Zilberstein. *Journal of AI Research* (JAIR), vol. 34, pages 89-132, February, 2009.
- **Analyzing myopic approaches for multi-agent communications.** Raphen Becker, Alan Carlin, Victor Lesser, and Shlomo Zilberstein. Computational Intelligence}, 25(1): 31—50, 2009.
- **Optimal and Approximate Q-value Functions for Decentralized POMDPs.** Frans A. Oliehoek, Matthijs T. J Spaan and Nikos Vlassis. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

References continued

- **Formal Models and Algorithms for Decentralized Decision Making Under Uncertainty.** Sven Seuken and Shlomo Zilberstein. *Journal of Autonomous Agents and Multi-Agent Systems* (JAAMAS). 17:2, pages 190-250, 2008.
- **Interaction-driven Markov games for decentralized multiagent planning under uncertainty**. Matthijs T. J. Spaan and Francisco S. Melo. *Proceedings of the Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 525--532, 2008.
- **Mixed integer linear programming for exact finite-horizon planning in Decentralized POMDPs**. Raghav Aras, Alain Dutech, and Francois Charpillet. In Int. Conf. on Automated Planning and Scheduling, 2007.
- **Value-based observation compression for DEC-POMDPs**. Alan Carlin and Shlomo Zilberstein.In *Proceedings of the Joint Conference on Autonomous Agents and Multi Agent Systems*, 2008.
- **Multiagent planning under uncertainty with stochastic communication delays**. Matthijs T. J. Spaan, Frans A. Oliehoek, and Nikos Vlassis. In *Int. Conf. on Automated Planning and Scheduling*, pages 338--345, 2008.
- **Improved memory-bounded dynamic programming for decentralized POMDPs**. Sven Seuken and Shlomo Zilberstein. *Proceedings of Uncertainty in Artificial Intelligence*, July, 2007
- **Memory-Bounded Dynamic Programming for DEC-POMDPs.** Sven Seuken and Shlomo Zilberstein. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligences* (IJCAI-07), January 2007
- **Networked Distributed POMDPs: A Synthesis of Distributed Constraint Optimization and POMDPs.** Ranjit Nair, Pradeep Varakantham, Milind Tambe and Makoto Yokoo. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (AAAI-05), Pittsburgh, PA, July, 2005.

References continued

- **A framework for sequential planning in multi-agent settings.** Piotr J. Gmytrasiewicz and Prashant Doshi. *Journal of Artificial Intelligence Research*, 24:49--79, 2005.
- **Bounded policy iteration for decentralized POMDPs.** Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005.
- **An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs**. Daniel Szer and Francois Charpillet. In *European Conference on Machine Learning*, 2005.
- **MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPS.** Daniel Szer, Francois Charpillet, Shlomo Zilberstein. *Proceedings of the 21st International Conference on Uncertainty in Artificial Intelligence* (UAI-05), Edinburgh, Scotland, July 2005
- **Reasoning about joint beliefs for execution-time communication decisions**. Maayan Roth, Reid G. Simmons and Manuela M. Veloso. *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems,* 2005
- **Dynamic Programming for Partially Observable Stochastic Games.** Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. *Proceedings of the 19th National Conference on Artificial Intelligence* (AAAI-04), 709-715, San Jose, California, July 2004
- **Approximate solutions for partially observable stochastic games with common payoffs.** Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. In *Proceedings of the Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.
- **An approximate dynamic programming approach to decentralized control of stochastic systems.** Randy Cogill, Michael Rotkowitz, Benjamin Van Roy and Sanjay Lall. In *Proceedings of the 2004 Allerton Conference on Communication, Control, and Computing*, 2004.
- ***Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis.*** Claudia V. Goldman and Shlomo Zilberstein. *Journal of Artificial Intelligence Research* Volume 22, pages 143-174, 2004.

References continued

- **Communication for improving policy computation in distributed POMDPs**. Ranjit Nair, Milind Tambe, Maayan Roth, and Makoto Yokoo. *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.
- **Planning, Learning and Coordination in Multiagent Decision Processes**. Craig Boutilier *Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK-96), Amsterdam, pp.195--210 (1996).*
- **Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings.** Ranjit Nair, David Pynadath, Makoto Yokoo, Milind Tambe and Stacy Marsella. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (IJCAI-03), Acapulco, Mexico, August, 2003.
- **The Complexity of Decentralized Control of Markov Decision Processes.** Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. *Mathematics of Operations Research*, 27(4):819-840, November 2002.
- **The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models.** David V. Pynadath and Milind Tambe. *Journal of Artificial Intelligence Research (JAIR)*, Volume 16, pp. 389-423. 2002.
- **Decision-theoretic control of planetary rovers**. Shlomo Zilberstein, Richard Washington, Daniels S. Bernstein, and Abdel-Illah Mouaddib. In *Plan-Based control of Robotic Agents*, volume 2466 of LNAI, pages 270--289. Springer, 2002.
- **Reinforcement learning for adaptive routing.** Leonid Peshkin and Virginia Savova. In Proceedings of the International Joint Conference on Neural Networks, 2002.
- **Decentralized control of a multiple access broadcast channel: Performance bounds.** James M. Ooi and Gregory W. Wornell. In *Proceedings of the 35th Conference on Decision and Control*, 1996.