How to Help our Planning Algorithms Succeed?

Shlomo Zilberstein

College of Information and Computer Sciences

University of Massachusetts Amherst

KR Workshop on Reasoning about Actions and Processes • October 28-29, 2018

Planning for Multiple Agents Under Partial Observability

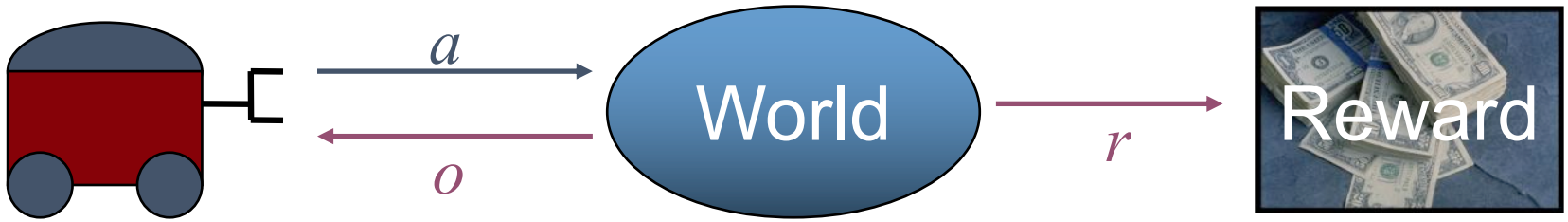
- ***Challenge:*** How to achieve intelligent coordination of a group of decision makers in spite of stochasticity and partial observability? How to handle the uncertainty about the domain, outcome of actions, and the knowledge, beliefs and intentions of the other agents?

- ***Objective:*** Find ways to make existing algorithms more effective by introducing additional structure into the problem or solution method.

Problem Characteristics

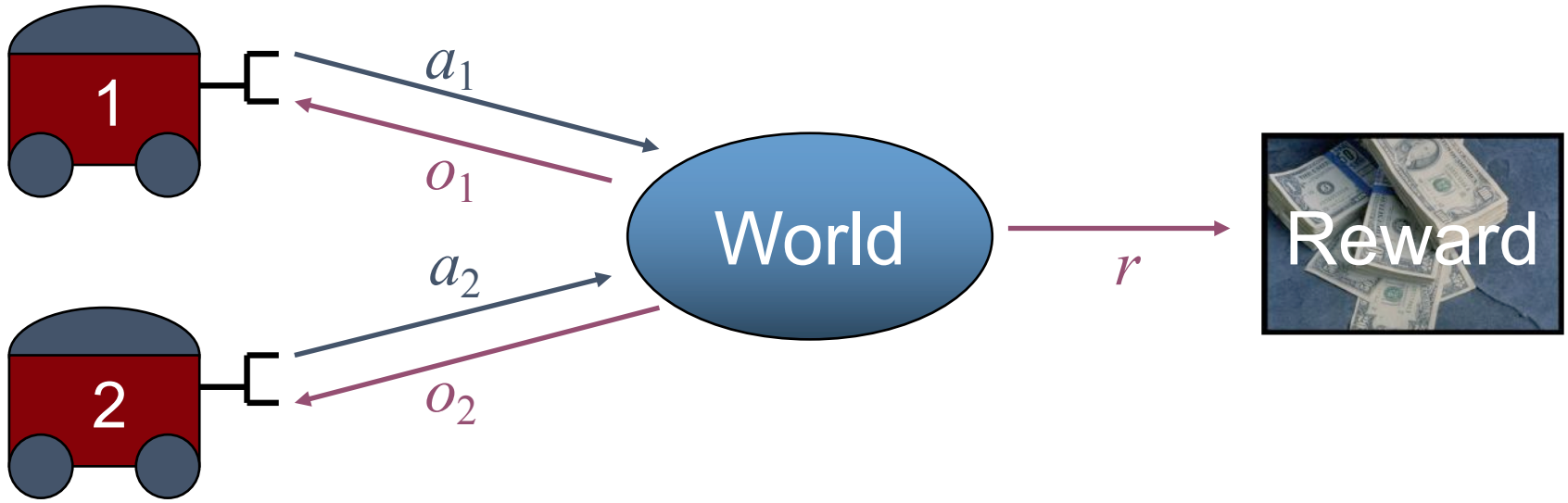
- A ***group*** of decision makers or agents interact in a stochastic environment
- Each problem instance involves a ***sequence*** of decisions over ***finite, indefinite, or infinite horizon***
- The change in the environment is determined ***stochastically*** by the ***current state*** and the ***set of actions*** taken by the agents
- Each decision maker obtains ***different observations*** and has ***different partial knowledge*** about the overall situation
- Focus on ***collaborative settings*** where decision makers share the ***same objectives***

Partially Observable MDP



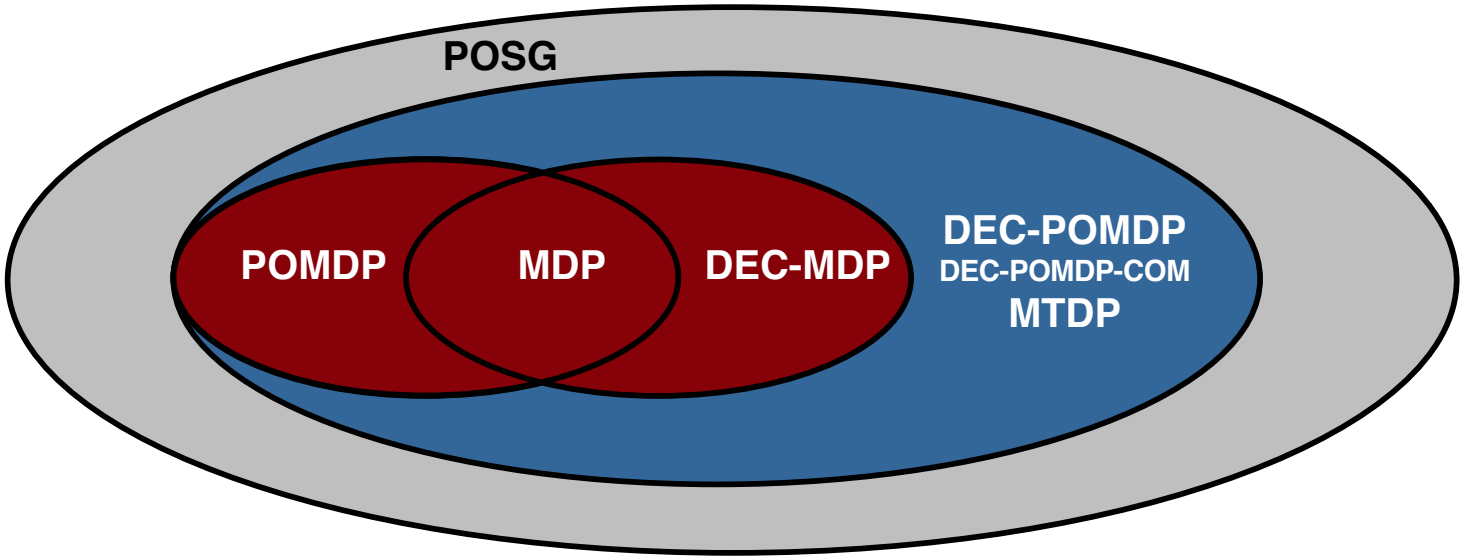
- Generalization of MDPs that was formulated in the 1960s [Astrom 65]
- The agent receives noisy observations of the underlying state
- Need to remember previous observations in order to act optimally
- More difficult, but there are DP algorithms [Smallwood & Sondik 73] and many modern solvers

Decentralized POMDP



- Generalization of MDP/POMDP involving multiple cooperating decision makers with different observation functions and actions

Formal Models



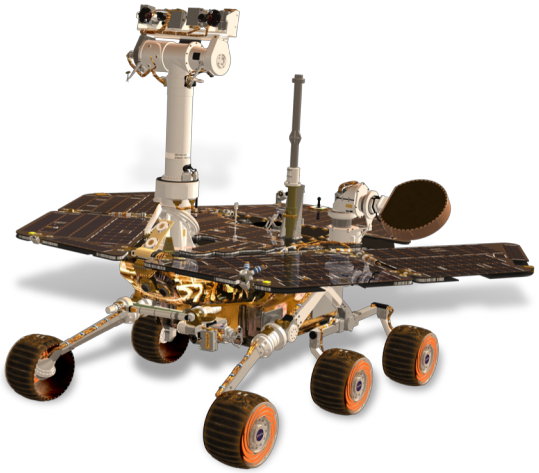
POSG = Partially-Observable Stochastic Game
DEC-POMDP-COM = DEC-POMDP with Communication
MTDP = Multiagent Team Decision problem

I-POMDP = Interactive POMDP

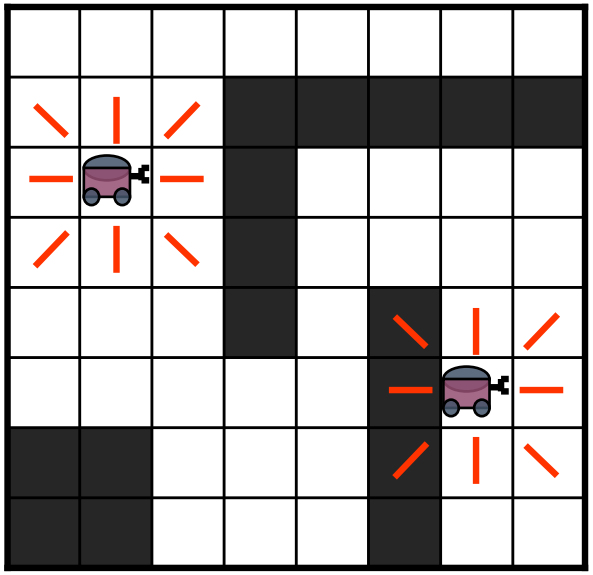
DEC-POMDPs

- A DEC-POMDP is defined by a tuple $\langle S, \vec{A}, P, R, \vec{\Omega}, O \rangle$, where
  - S is a finite set of domain states, with initial state s_0
  - $\vec{A} = \{A_1, A_2, \ldots, A_n\}$ are finite action sets
  - $P(s, \vec{a}, s')$ is a state transition function
  - $R(s, \vec{a})$ is a reward function
  - $\vec{\Omega} = \{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ are finite observation sets
  - $O(\vec{a}, s', \vec{o})$ is an observation function
- Agents have different partial knowledge about the domain; choose actions based on their private observations.

Applications

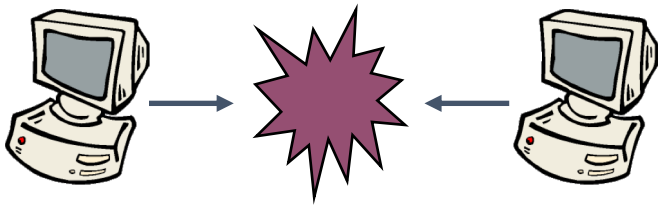


Space exploration rovers



Coordination of mobile robots

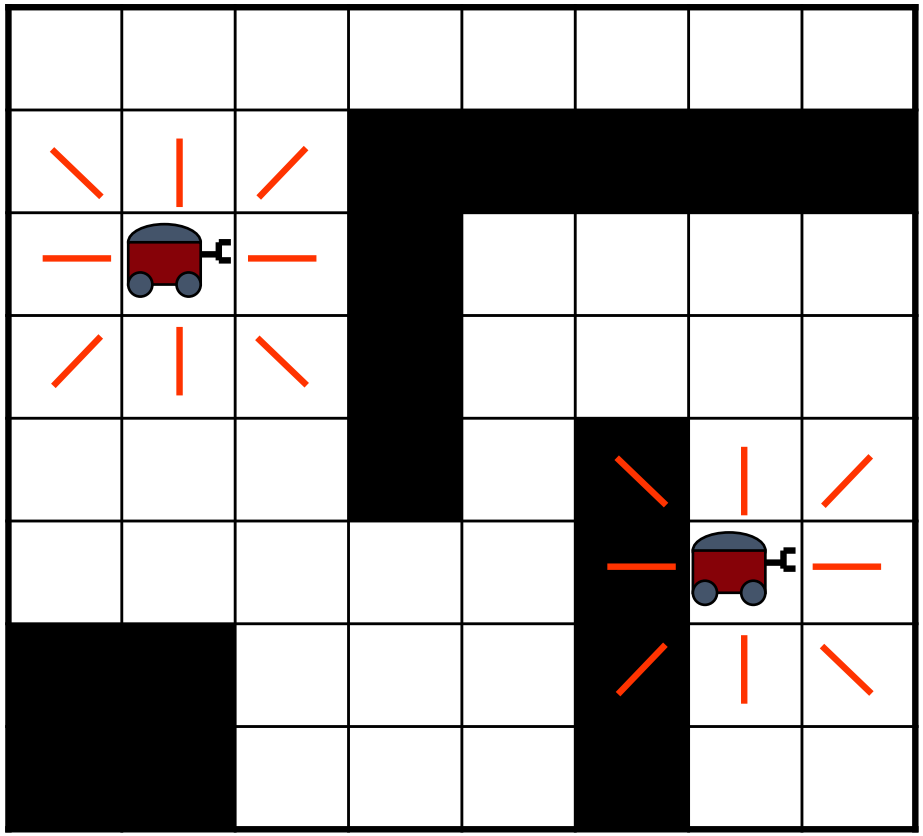
Decentralized detection of hazardous weather events



Multi-access broadcast channels

e.g. Mobile Robot Coordination

Bernstein, Hansen & Zilberstein, IJCAI 2005



States: grid cell pairs

Actions: ↑,↓,←,→

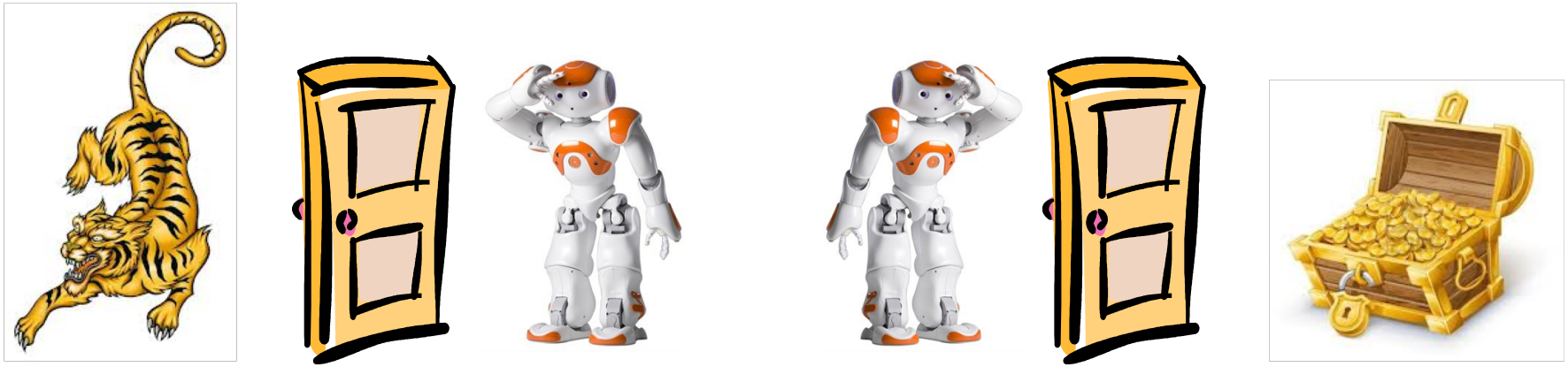
Transitions: noisy

Goal: meet quickly

Observations: red lines

e.g. Multiagent Tiger

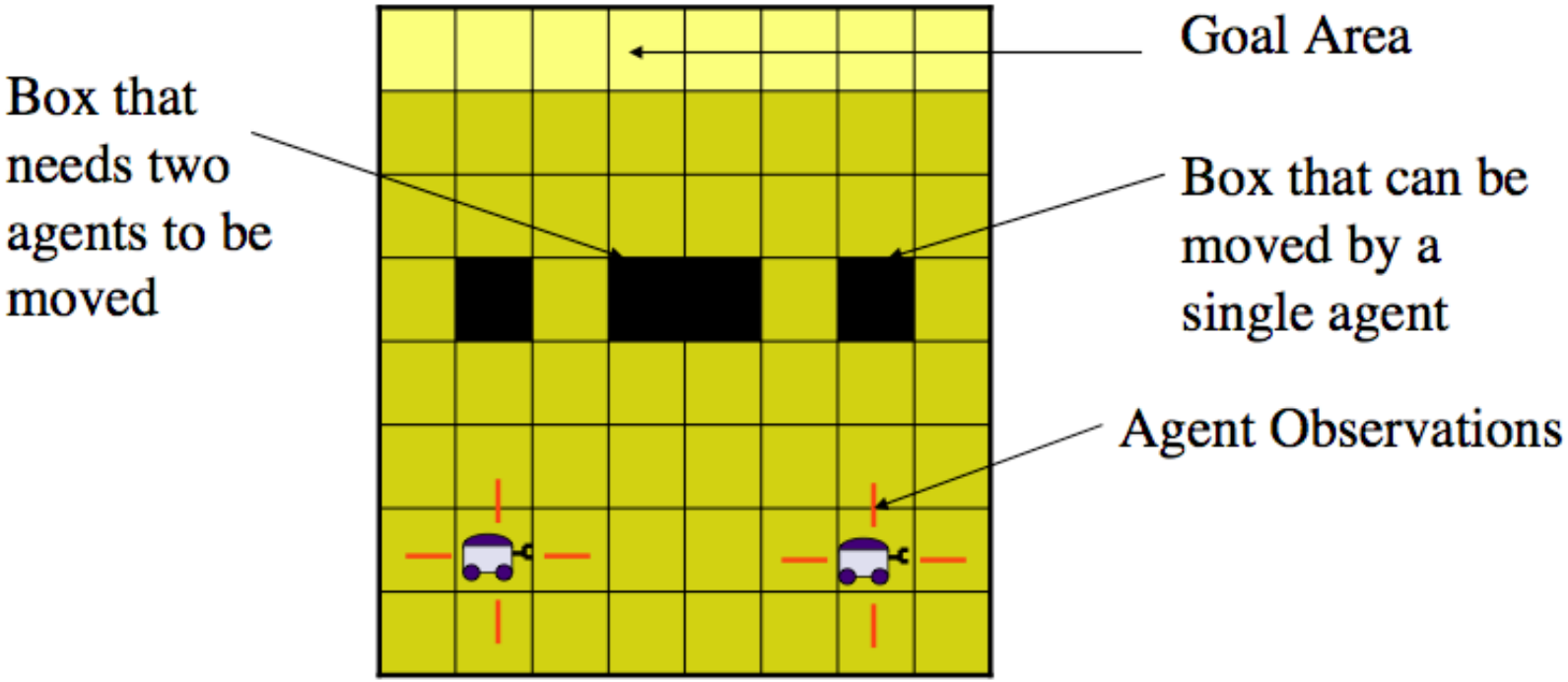
Nair, Tambe, Yokoo, Pynadath & Marsella, 2003



- Two agents try to locate tiger and get treasure
- Each agent may open one of the doors or listen
- Listening provides a noisy observation
- Large penalty for opening door leading to tiger; Large reward for cooperation (choosing same action) and for getting the treasure.

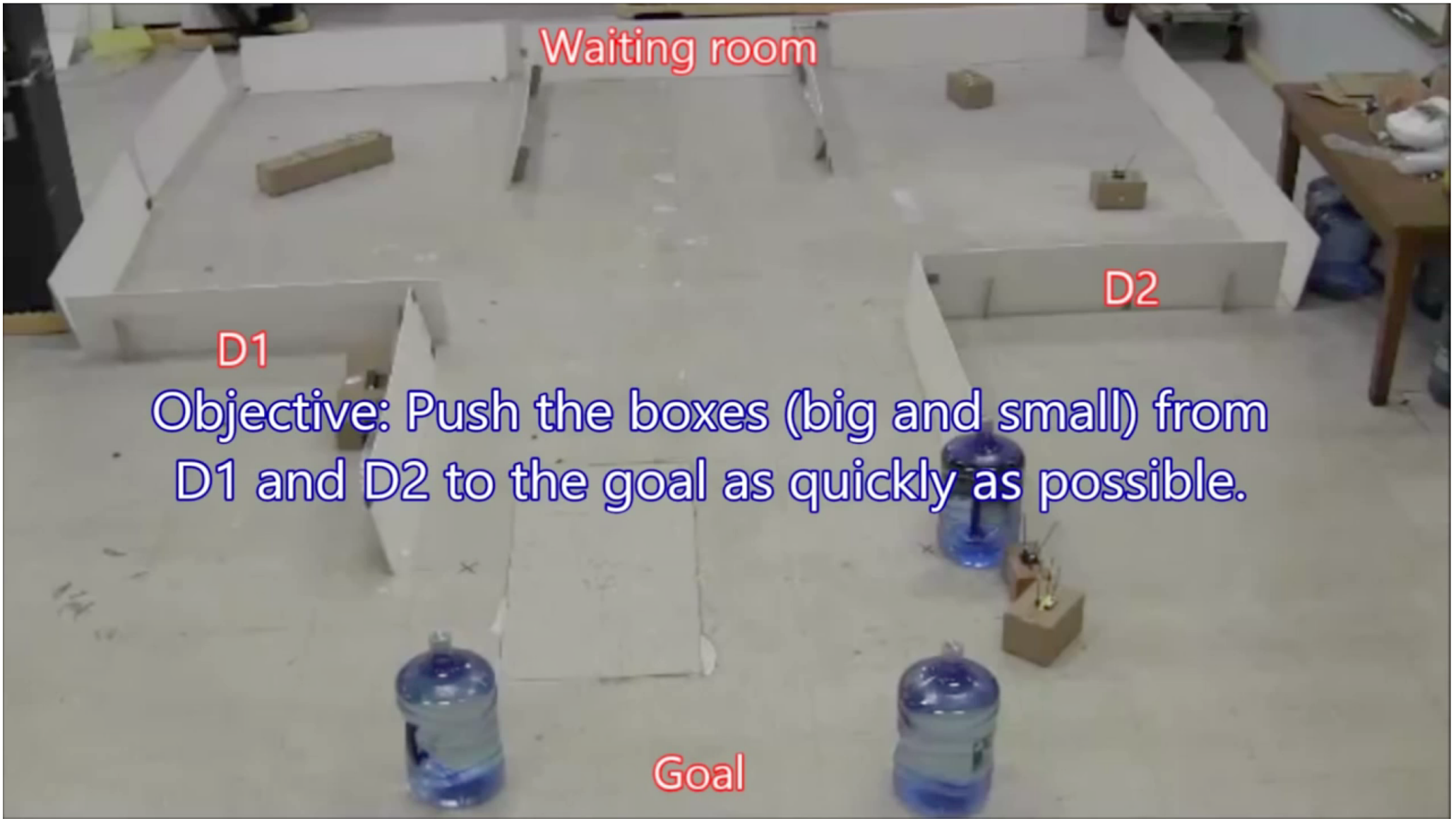
e.g. Cooperative Box-Pushing

Seuken & Zilberstein, UAI 2007



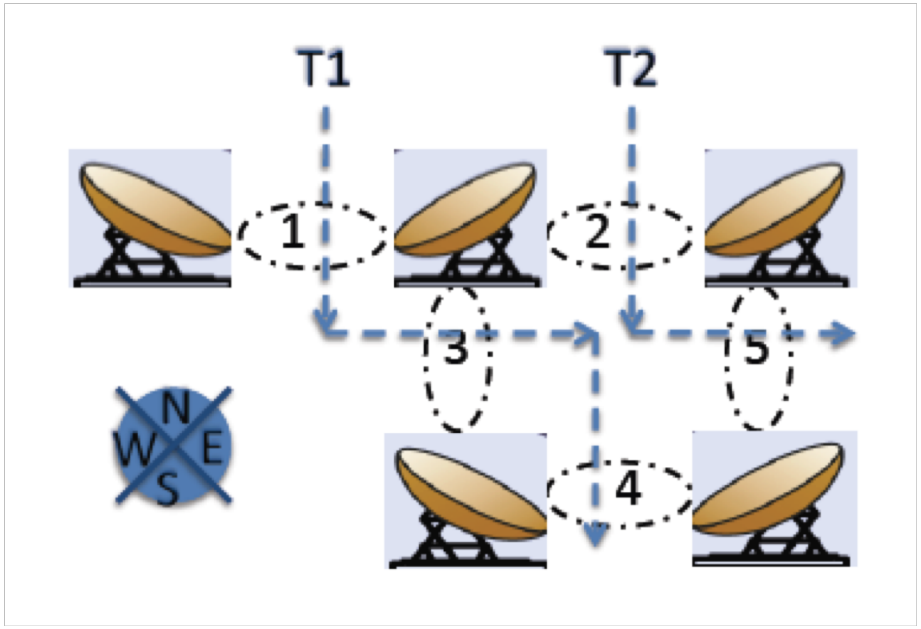
Goal: push as many boxes as possible to goal area; larger box has higher reward, but requires two agents to be moved.

Box Pushing Demo



e.g. Sensor Network

Nair et al. 2005; Kumar and Zilberstein, 2009

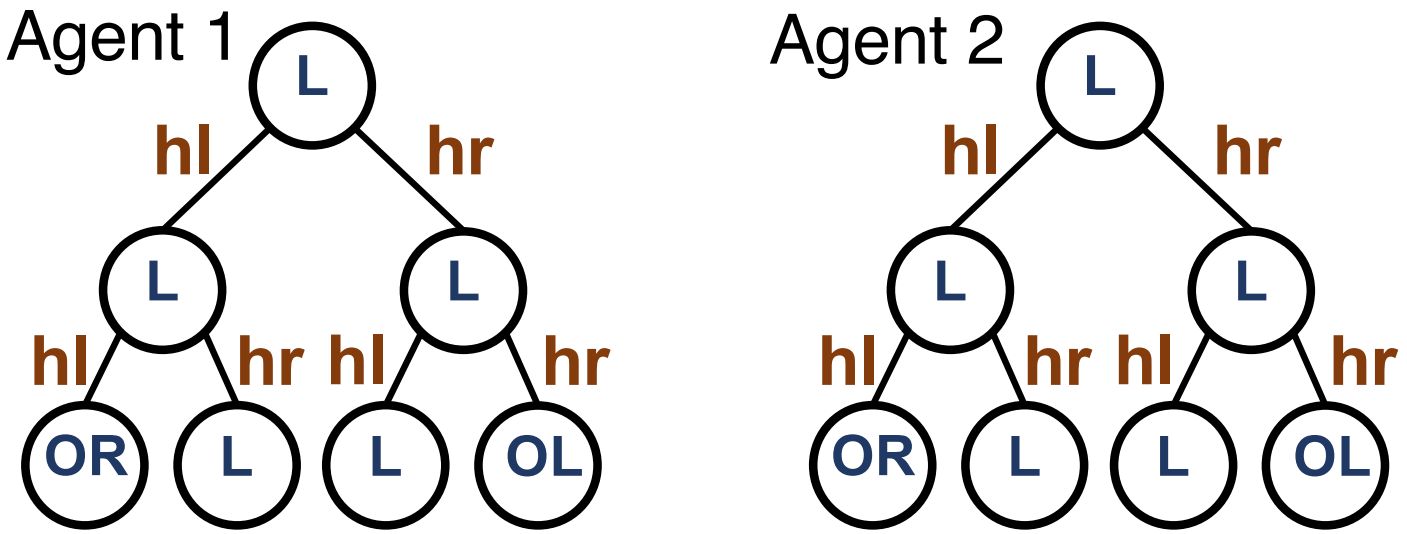


- Multiple sensors track a moving target
- To localize target two neighboring sensors must track it at the same time

Solving DEC-POMDPs

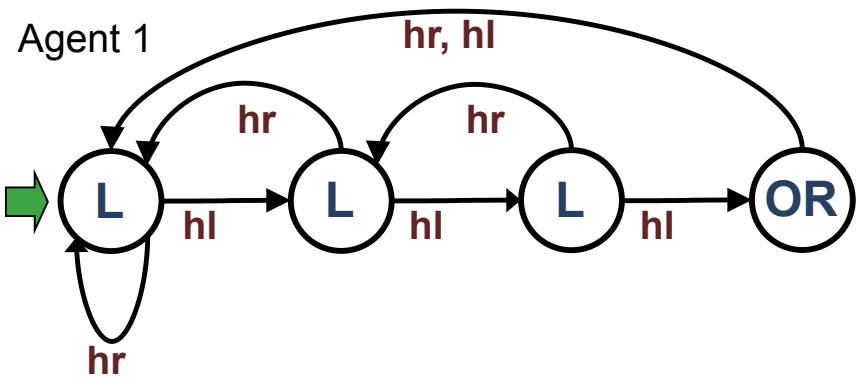
- Each agent's behavior is described by a ***local policy*** δ_i
- Policy can be represented as a mapping from
  - Local ***observation sequences*** to ***actions***; or
  - Local ***memory states*** to ***actions***
- Actions can be selected ***deterministically*** or ***stochastically***
- Goal is to ***maximize expected reward*** over a finite horizon or discounted infinite horizon

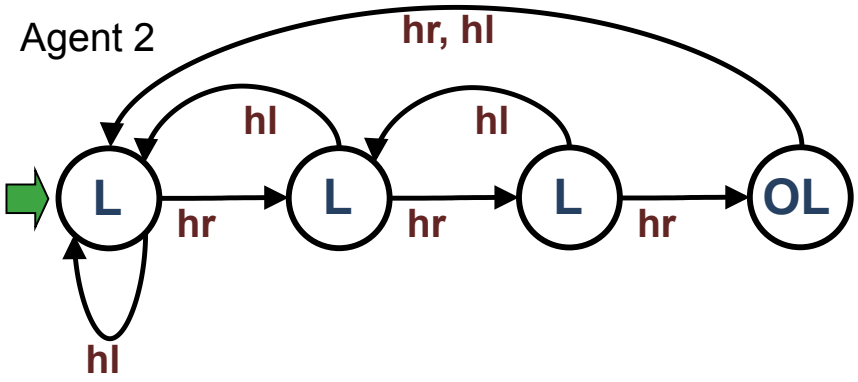
Policy Trees



- Optimal policy trees for the multiagent tiger problem with horizon 3. The policy trees of both agents are the same
- Each node is labeled with an action and each edge is labeled with an observation
- Suitable for finite-horizon problems

Finite-State Controllers





- Optimal four-node deterministic controllers for the multiagent tiger problem
- Each node is labeled with an action and each transition is labeled with an observation
- Suitable for infinite horizon problems

Key Questions

- What is the complexity of DEC-POMDPs?
- Are they significantly harder than POMDPs? Why?
- What domain features affect the complexity and how?
- Is optimal dynamic programming possible?
- Can dynamic programming be made practical?
- How to treat communication?
- How to develop more scalable solution methods?
  (with respect to #agents, #states, #observations)

Complexity Results for (PO)MDPs

Finite Horizon

MDP	P-complete (if $T < \|S\|$)	Papadimitriou & Tsitsiklis 87
POMDP	PSPACE- complete (if $T < \|S\|$)	Papadimitriou & Tsitsiklis 87

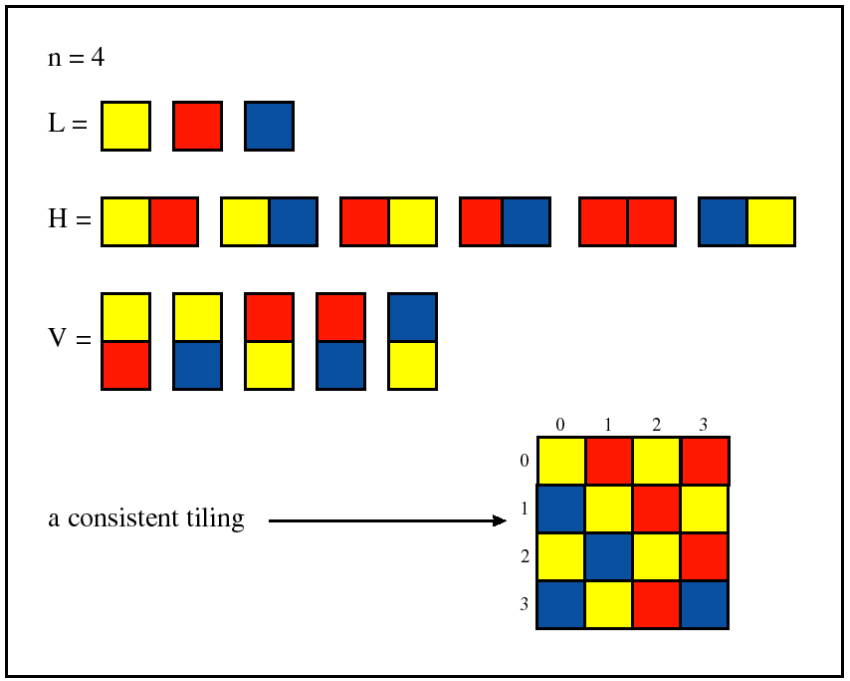
Infinite-Horizon Discounted

MDP	P-complete	Papadimitriou & Tsitsiklis 87
POMDP	Undecidable	Madani et al. 99

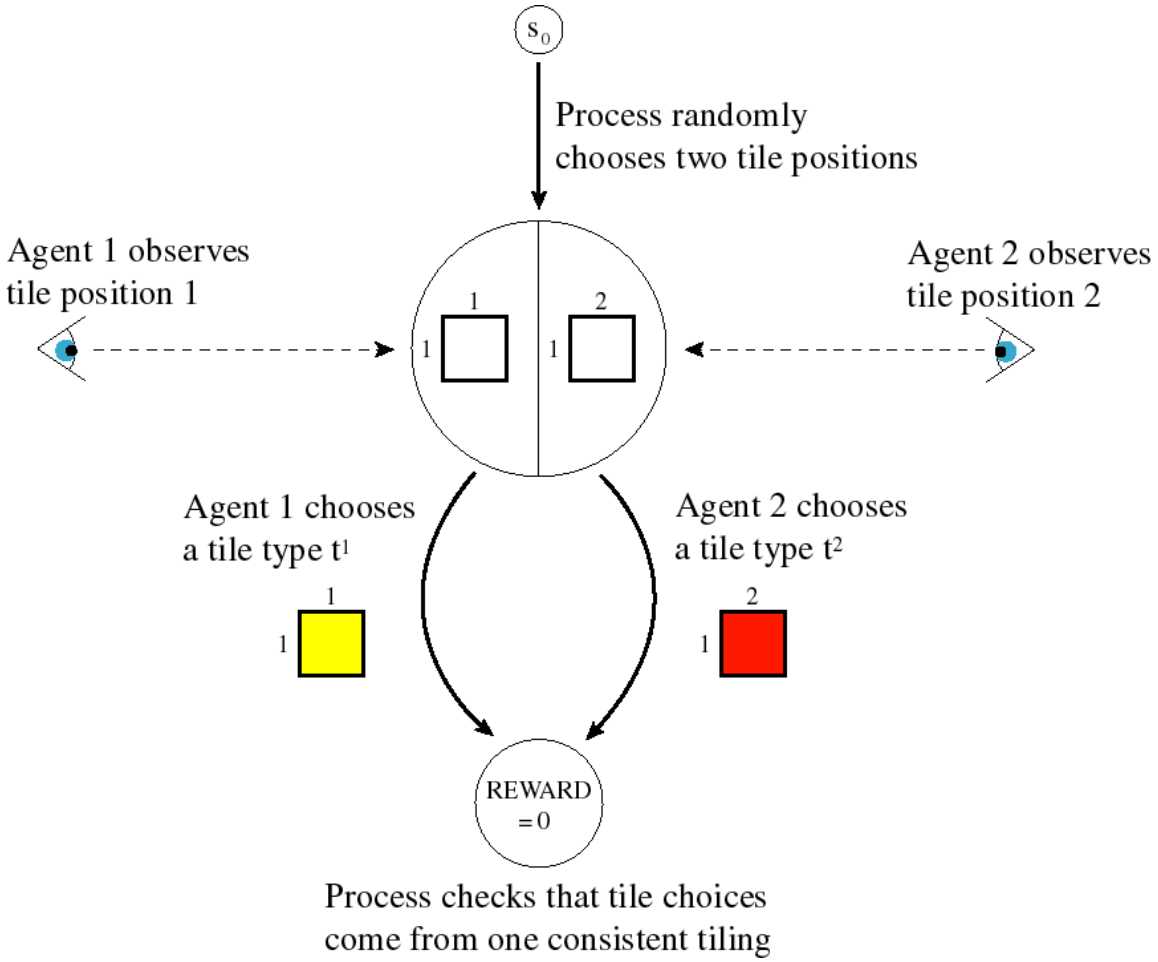
How Hard are DEC-POMDPs?

Bernstein, Givan, Immerman & Zilberstein, UAI 2000, MOR 2002

- ***Static*** version where a ***single*** set of decisions is made in response to a ***single*** set of observations, was shown to be NP-hard [Tsitsiklis and Athan, 1985]
- Two-agent DEC-POMDPs ∈ **NEXP-hard** by reduction from **TILING**



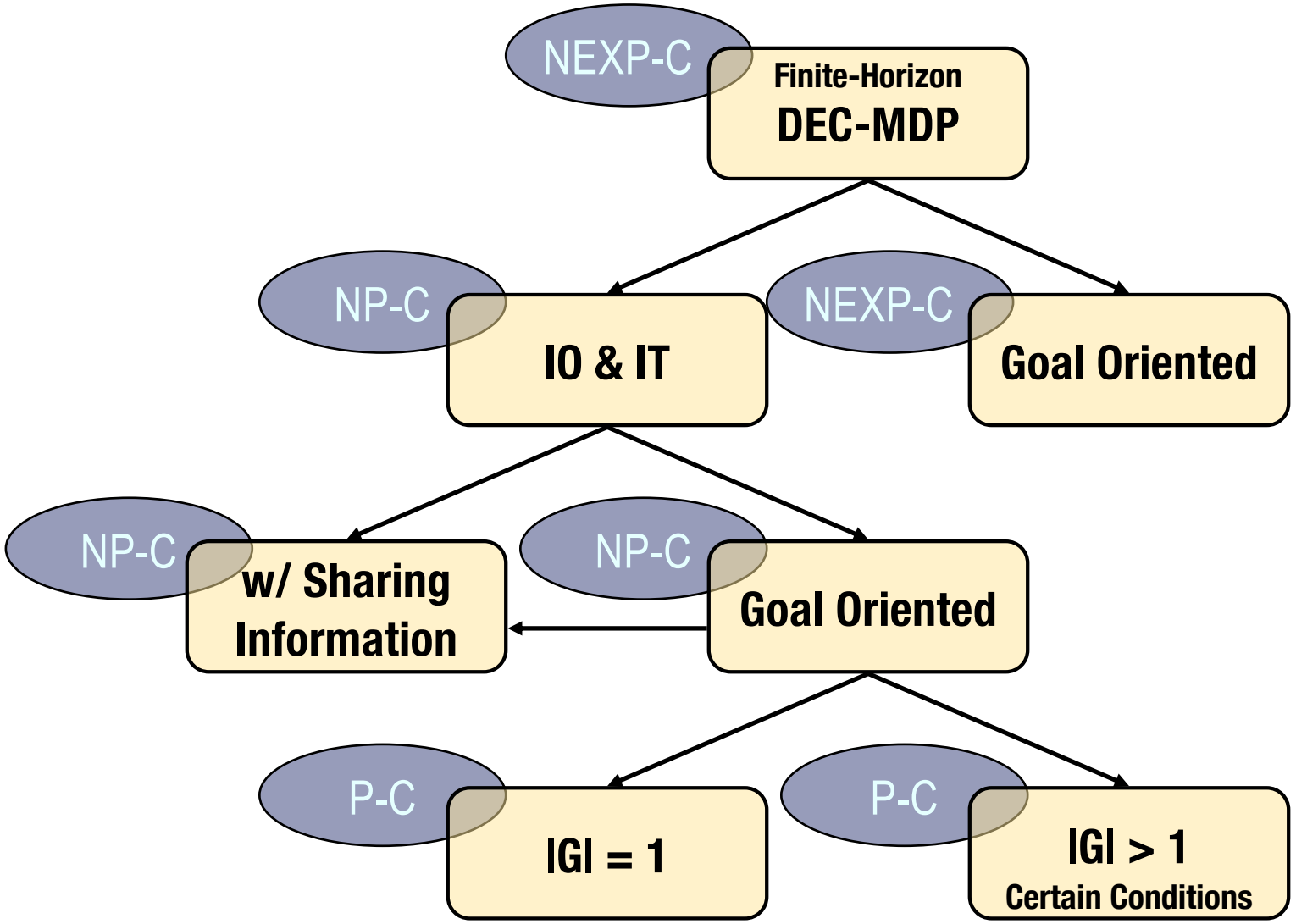
DEC-POMDP is NEXP-hard



∃ policy with
expected reward 0
⇔
∃ consistent tiling

Complexity of Sub-Classes

Goldman & Zilberstein, JAIR 2004



Algorithms for DEC-POMDPs

- Unclear how to define a standard ***belief-state*** for a DEC-POMDP without fixing the policies of the other agents or having beliefs about them
- ***Value iteration*** does not generalize to the infinite-horizon case
- ***Policy iteration*** for POMDPs can be generalized [Hansen 98, Poupart & Boutilier 04]
- Basic idea: Representing local policies using ***stochastic finite-state controllers*** and defining a set of controller transformations that guarantee convergence

Policies as Controllers

- Finite state controller represents each policy
  - Fixed memory
  - Randomness used to offset memory limitations
  - Action selection, $\psi : Q_i \rightarrow \Delta A_i$
  - Transitions, $\eta : Q_i \times A_i \times O_i \rightarrow \Delta Q_i$
- Value of two-agent joint controller given by the Bellman equation:

$$V(q_1,q_2,s) = \sum_{a_1,a_2} P(a_1 \mid q_1)P(a_2 \mid q_2)\Big[R(s,a_1,a_2) + \gamma \sum_{s'} P(s' \mid s,a_1,a_2) \sum_{o_1,o_2} O(o_1,o_2 \mid s',a_1,a_2) \sum_{q_1',q_2'} P(q_1' \mid q_1,a_1,o_1)P(q_2' \mid q_2,a_2,o_2)V(q_1',q_2',s')\Big]$$

Finding Optimal Controllers

- How can we ***search the space*** of possible joint controllers?
- How do we set the parameters of the controllers to ***maximize value***?
- Deterministic controllers can be found using traditional search methods such as BFS
- Stochastic controllers present a continuous optimization problem
- ***Key question***: how to best use a limited amount of storage to optimize value

Nonlinear Optimization Approach

Amato, Bernstein & Zilberstein, UAI 2007

- Key idea: Modeling the problem as a non-linear program (NLP)
- Consider node values (as well as controller parameters) as a variables
- The NLP can take advantage of an initial state distribution when it is given
- Improvement and evaluation all in one step (equivalent to an infinite lookahead)
- Additional constraints maintain valid values

NLP Representation

Variables:

$$x(\vec{q},\vec{a}) = P(\vec{a} \mid \vec{q}), \quad y(\vec{q},\vec{a},\vec{o},\vec{q}') = P(\vec{q}' \mid \vec{q},\vec{a},\vec{o}), \quad z(\vec{q},s) = V(\vec{q},s)$$

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

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

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

Additional linear constraints:

- ensure controllers are independent
- all probabilities sum to 1 and are non-negative

Independence Constraints

- Independence constraints guarantee that action selection and controller transition probabilities for each agent depend only on local information
- Action selection independence:

$$\forall a_i, \vec{q} \sum_{a_{-i}} x(\vec{q}, \vec{a}) = \sum_{a_{-i}} x(q_i, q^f_{-i}, \vec{a})$$

- Controller transition independence:

$$\forall \vec{a}, \vec{q}, \vec{o}, q'_i \sum_{q'_{-i}} y(\vec{q}, \vec{a}, \vec{o}, \vec{q'}) = \sum_{q'_{-i}} y(q_i, q^f_{-i}, a_i, a^f_{-i}, o_i, o^f_{-i}, \vec{q'})$$

Probability Constraints

- Probability constraints guarantee that action selection probabilities and controller transition probabilities are non negative and that they add up to 1:

$$\forall q_i \sum_{\vec{a}} x(q_i, q^f_{-i}, \vec{a}) = 1 \text{ and } \forall q_i, o_i, a_i \sum_{\vec{q'}} y(q_i, q^f_{-i}, a_i, a^c_{-i}, o_i, o^f_{-i}, \vec{q'}) = 1$$

$$\forall \vec{q}, \vec{a}\ x(\vec{q}, \vec{a}) \geq 0 \quad \text{and} \quad \forall \vec{q}, \vec{o}, \vec{a}\ y(\vec{q}, \vec{a}, \vec{o}, \vec{q'}) \geq 0$$

(Superscript *f*'s represent arbitrary fixed values)

Optimality

Theorem: *An optimal solution of the NLP results in optimal stochastic controllers for the given size and initial state distribution.*

- Advantages of the NLP approach:
  - Efficient policy representation with fixed memory
  - NLP represents optimal policy for given size
  - Takes advantage of known start state
  - Easy to implement using off-the-shelf solvers
- Limitations:
  - Difficult to solve optimally

Six Aspects of MAS that DEC-POMDP Solvers Optimize "on the fly"

1. **Internal Memory Structure**
   — *What is worth memorizing using bounded memory?*
2. **Hierarchical Decomposition of the Domain**
   — *What are the atomic decisions at each level of abstraction?*
3. **Information Gathering Actions**
   — *What should an agent do to reduce uncertainty efficiently?*
4. **Communication Protocol Design**
   — *What messages to use and what should they mean?*
5. **Organizational Design**
   — *What role should each agent have?*
6. **Social Norms and Commitments**
   — *What social norms and commitments can increase efficiency?*

Is this too much to expect?

How to Help our Planning Algorithms Succeed?

1. Internal Memory Structure

- Existing planning algorithms don't assign explicit meaning to internal memory states of the FSC
- Memory states have implicit meaning that can be derived by analyzing the policy
- In contrast, ML algorithms often use predefined (sometimes handcrafted) features to perform well
- Automatic feature induction is usually treated as a separate aspect of the problem
- People can often identify useful domain feature for coordination and, when they cannot, they can redesign the environment to provide such features

What Can be Memorized?

What can be memorized in the context of DEC-POMDPs?

– Generally, can memorize features of histories

– Let $h^i = \langle a^1, o^1, a^2, o^2, ..., a^i, o^i \rangle$

– Let $f : H \rightarrow M$ map histories to memory states

– For f to be feasible, need a Markovian update function:

such that: $f(h^{i+1}) = f(h^i) \circ \langle a^{i+1}, o^{i+1} \rangle$

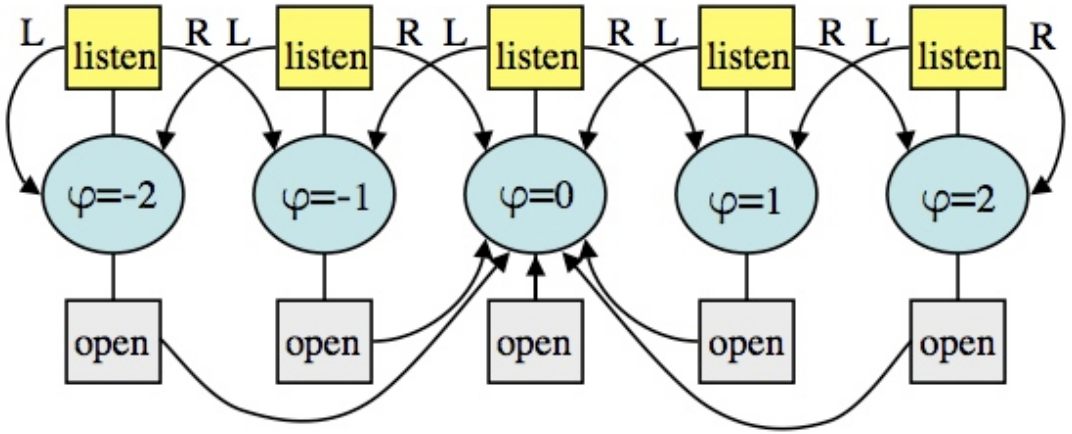
Examples:

- Reactive policy where $f(h^{i+1}) = o^{i+1}$
- Remember if observation o_j has been seen
- Any regular expression over Ω^*

Attribute-based Controllers

Amato & Zilberstein, ICAPS MAP WS 2008

- Designing controllers where each state has predefined semantics derived from the history of observations
- Example: Decentralized Tiger



BFS	DEC-BPI	NLP	Attribute Mapping
-14.115 (3 nodes, 12007s)	-52.633 (11 nodes, 102s)	-1.088 (19 nodes, 6173s)[1]	4.594 (7 nodes, 127s)

History-Based Controller Design

Kumar & Zilberstein, ICAPS 2015

- When optimizing policies represented as FSCs, it's hard to determine how much memory is needed and what's worth memorizing
- Start with a fixed-size controller, for example a reactive controller with one node per available observation
- Optimize this controller (in this case using a dual mixed integer linear program (MIP) for FSCs for POMDPs)
- Developed a history-based controller design – associating clear history-related properties with each controller node

History-Based Controller Design

Kumar & Zilberstein, ICAPS 2015

- Use an entropy-based node splitting criteria
- Somewhat similar to node splitting strategy proposed in [Poupart, Lang, and Toussaint 11] based on state-splitting while using EM for HMMs [Siddiqi, Gordon, and Moore 07]
- Novelty is the development of an entropy-based heuristic that uses the readily available information from the dual MIP formulation
- Key intuition is that we quantify the uncertainty about the world state associated with every HBC node, then split the node with the highest uncertainty

History-Based Controller Design

Kumar & Zilberstein, ICAPS 2015

- Results

	dualMIP			IsoBnB		NLP	
	Reactive	MaxEnt					
Problem	Qual.	Qual.	Time	Qual.	Time	Qual.	Time
tiger.95 ($\|S\|=2$ $\|A\|=3, \|O\|=2$) HSVI = 19.3	-20(opt)	19.3	0.44	19.3	1.4	-0.63	3.79
hallway ($\|S\|=60$ $\|A\|=5, \|O\|=21$) HSVI = 0.52	0.38(0.42)	0.46	3345	0.19	8170	0.47	362
hallway2 ($\|S\|=93$ $\|A\|=5, \|O\|=17$) HSVI = 0.35	0.24(0.29)	0.28	5700	0.15	5616	0.28	420
machine ($\|S\|=256$ $\|A\|=4, \|O\|=16$) HSVI = 63	33.2(42.1)	62.54	950	62.6	41200	62.4	2640
tag ($\|S\|=870$ $\|A\|=5, \|O\|=30$) HSVI = -6.37	-17.2(-2.2)	-7.81	6000	?	?	-13.94	5596

Table 3: Quality and runtime (in sec.) comparisons for different controller optimization algorithms. 'Time' for 'MaxEnt' includes the time for optimizing the reactive controller.

Problem	tiger	hallway	hallway2	machine	tag
$\|Y\|+1$	3	22	18	17	31
Size	10	36	35	20	56

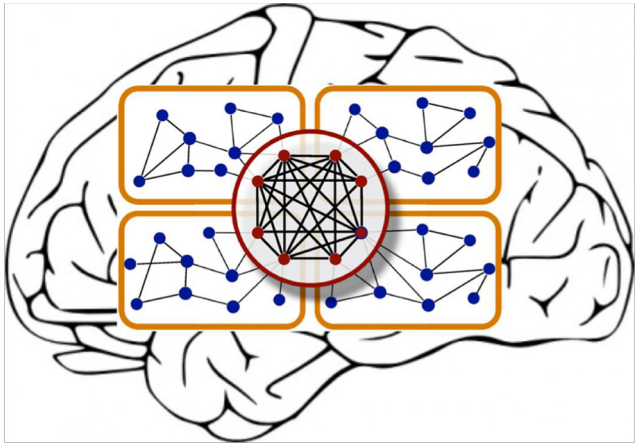
Table 4: Final controller size, including both the size of the initial reactive controller ($=|Y|+1$) and the number of nodes added by MaxEnt.

Informative Observations

- Observations should not be limited to state evidence, but also include indicators of goals, intentions, etc.
- Design observations to facilitate effective coordination
- Examples: vehicle signals; traffic signals; semaphores
- Critical aspects of coordination could be solved by design

2. Hierarchical Decomposition

- Planning can be generally simplified using a hierarchical approach with limited scope of decisions per level
- Coordination in particular could be simplified using abstraction



- In exiting approaches, domain-level actions and coordination actions are intermingled, often all at one level of decision making
- That makes both planning and coordination extremely complicated

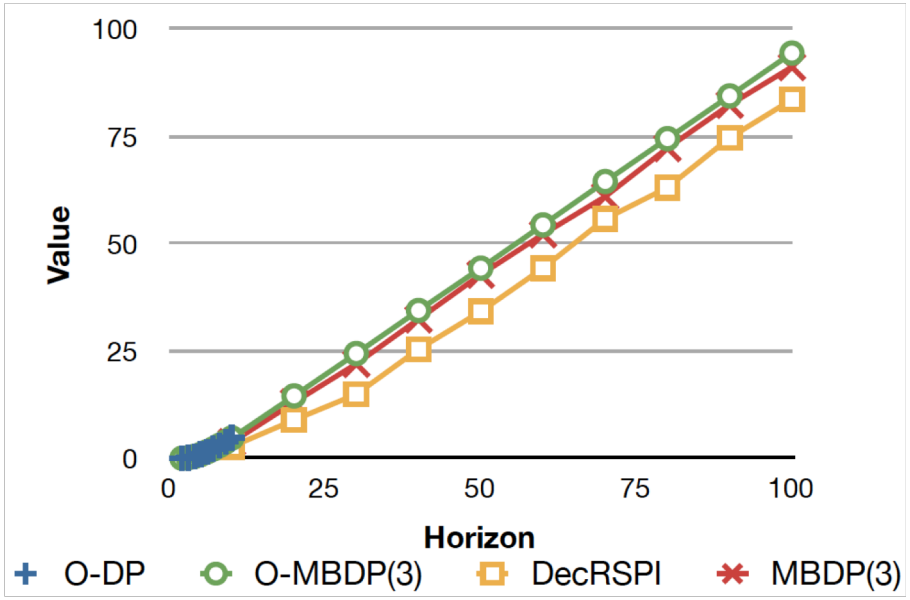
Many Relevant Techniques

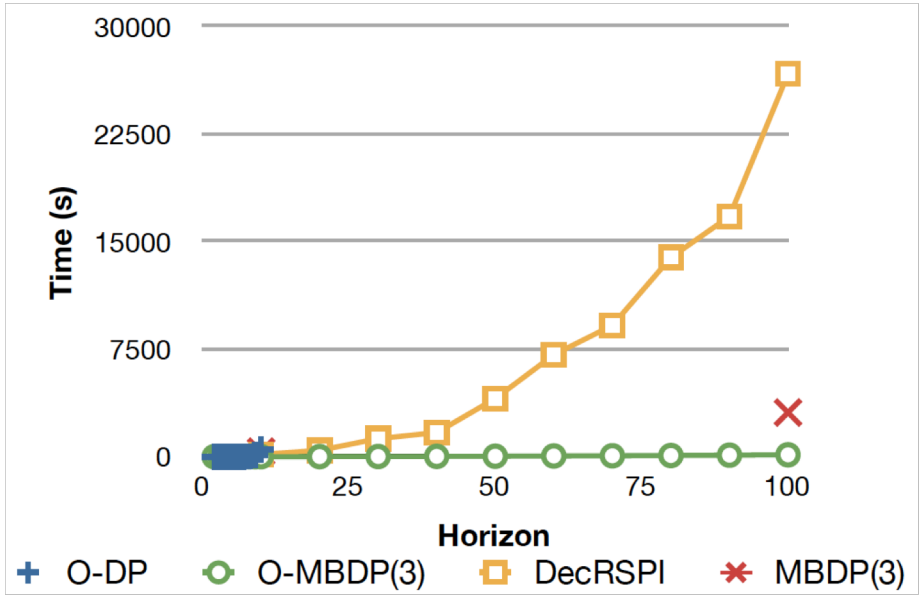
- Hierarchical reinforcement learning
  - Options; HAMs, MAXQ
- Hierarchical factored MDPs and POMDPs
  - [Dietterich; Parr & Russell; Guestrin & Gordon; Hansen]
- Hierarchical task networks
  - [Nau et al.; Erol; Tate; Fox]
- Planning with macro actions or complex actions
  - [McIlraith et al.; Marthi, Russell & Wolfe]
- Service composition methods
  - [McIlraith et al.; Traverso et al.]
- Behavior-based robotics

Planning with Macro Actions

Amato, Konidaris & Kaelbling, AAMAS 2014

- Extended DP and MBDP to work with macro actions
- For "meeting in a grid" defined two options for each agent: each one moving the agent to one of the two goal corners.
- Results in better value and significant time savings

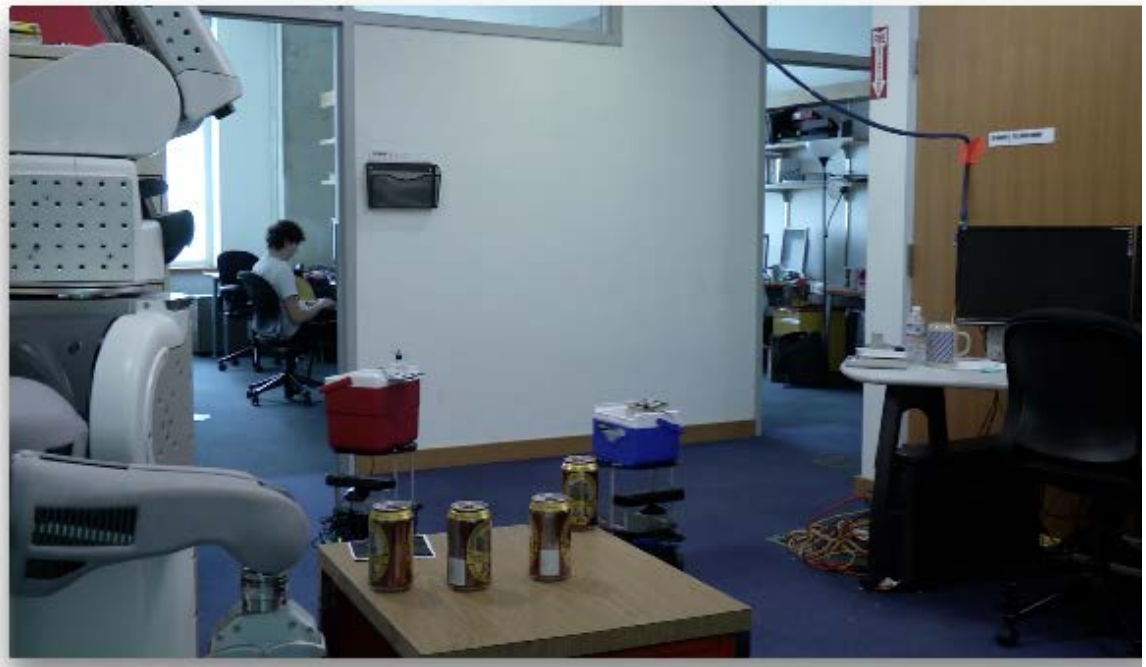




e.g. Bartender and Waiters

Amato et al., RSS 2015

- A multi-robot problem modeled after waiters gathering drinks and delivering them to different rooms
- Solved using the MacDec-POMDP model, which adds macro actions to Dec-POMDPs



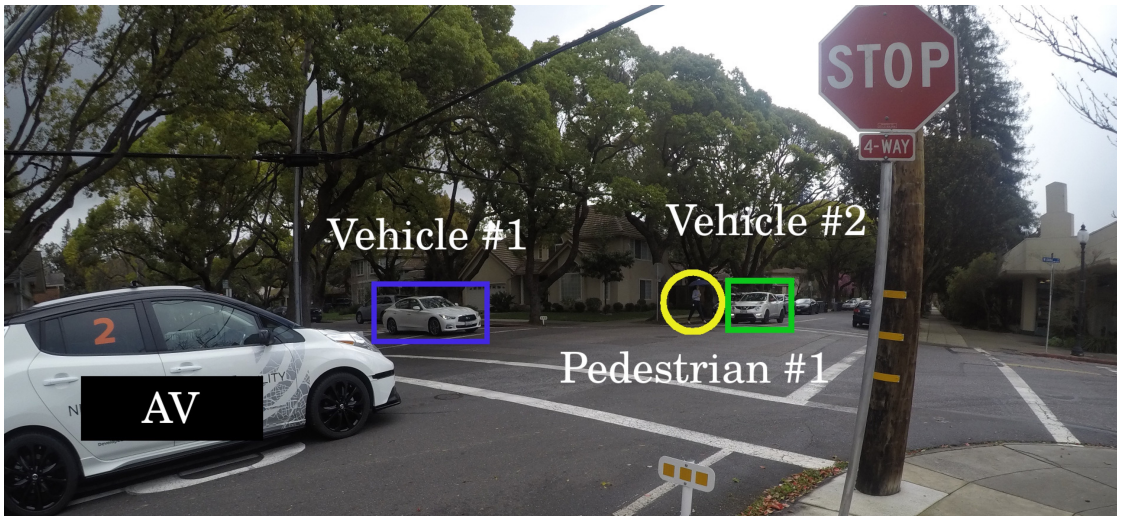
Using Macro Actions

Amato et al., RSS 2015

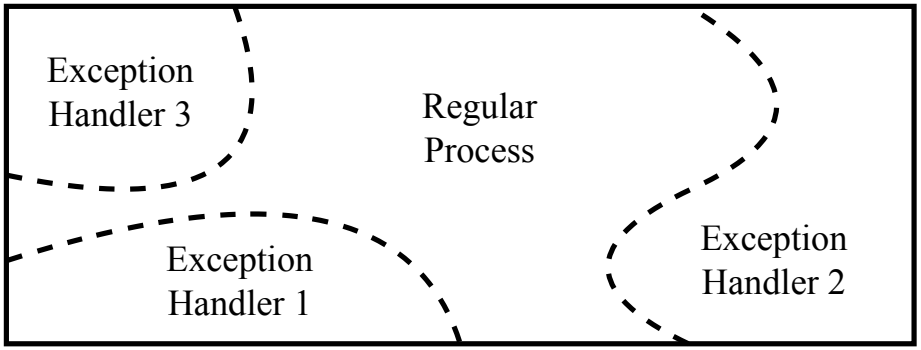
- Macro actions for the waiters:
  - **ROOM_N**: Go to room *n*, observe orders and deliver drinks
  - **BAR**: Go to the bar and observe current status of the bartender
  - **GET_DRINK**: Obtain a drink from the bartender
- The MDHS planner automatically generates solutions based on the macro-actions and high-level problem description
- Can solve problems in a few minutes that are intractable otherwise
- Open questions remain regarding option generation

Hierarchical Metareasoning for Exception Recovery

- Developed for handling exception in autonomous driving



- Using belief space partitioning with exception handlers

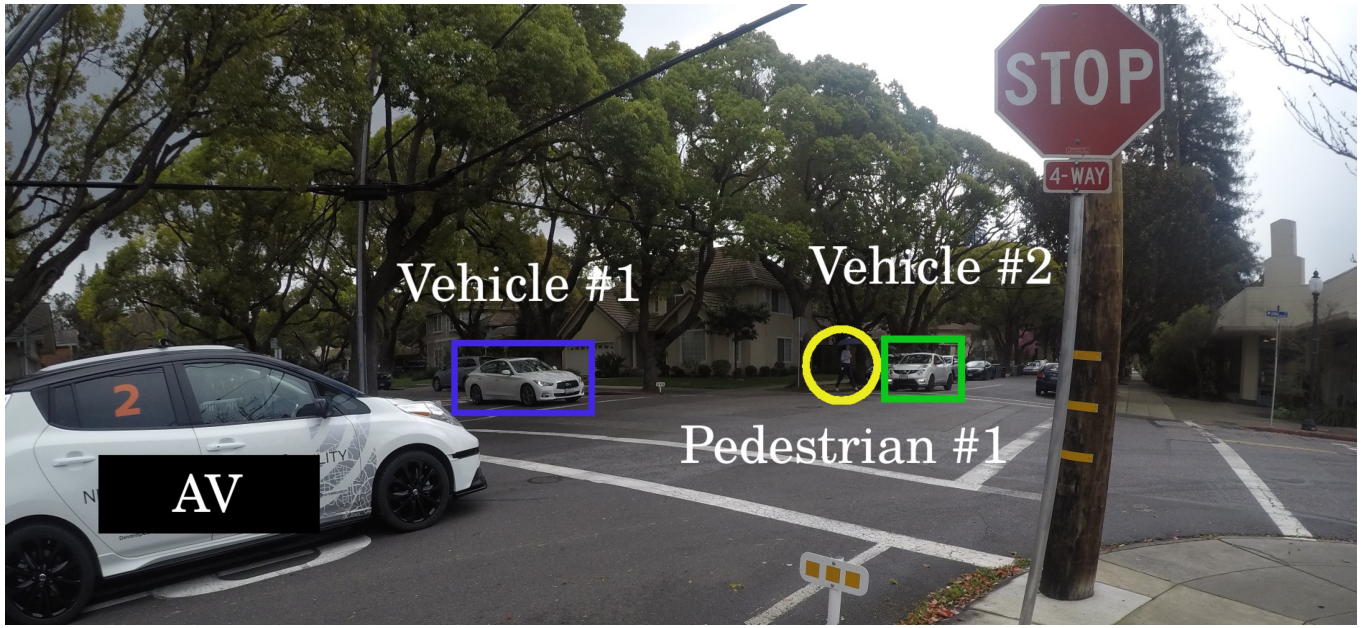


3. Information Gathering Actions

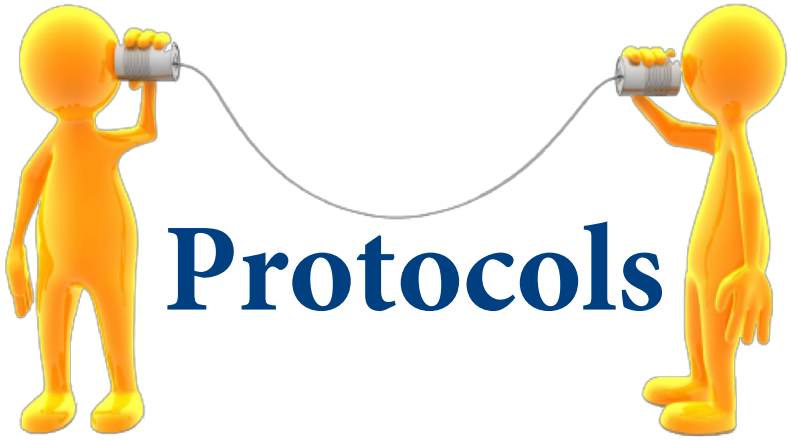
- Planner does not distinguish between three benefits of actions:
  - Contribution to goal achievement
  - Contribution to information gathering
  - Contribution to communication
- The value of an action reflects all these contributions
- Hard to reason explicitly about information gathering actions
- Each action could have both useful consequences and also unintended negative consequences along each dimension
- Beneficial to create “pure” information gathering actions

Autonomous Vehicles Example

- Crossing intersections presents complex interactions with multiple, simultaneously encountered entities
- Edging could be used to start crossing, gain better visibility, or convey intention (or all three)
- Planning is easier when actions are not so overloaded with consequences



4. Communication Protocols



- Supposed that ***one bit*** can be exchanged between agents,
  - What should each value mean?
  - When should a message be sent if communication is limited?
  - Should the meaning change over time based on context?
- Interestingly, these questions are ***answered implicitly*** by a standard DEC-POMDP policy
- Work on explicit communication has focused on sharing knowledge (some or all local observations), but not so much on language and protocol design

Communication-Based Decomposition

Goldman & Zilberstein, JAIR 2008

- Introduce mechanisms to decouple domain actions from communication actions for Dec-MDPs
- Compute multiagent macro actions that include no communication, but always end with communication
- For Dec-MDP, communication involve sharing the most recent observations, which provide full observability
- Solving the resulting Dec-SMDP-Com model is equivalent to solving an MMDP rather a Dec-MDP (lower complexity!)

Myopic Communication Strategies

Becker et al., CI 2009

- When communication is costly, can use a myopic rule to decide when to communicate
- Basic myopic rule can be evaluated in polynomial time

Definition 2. The Value of Communication (VoC) is the difference between the expected value when communicating and the expected value for remaining silent.

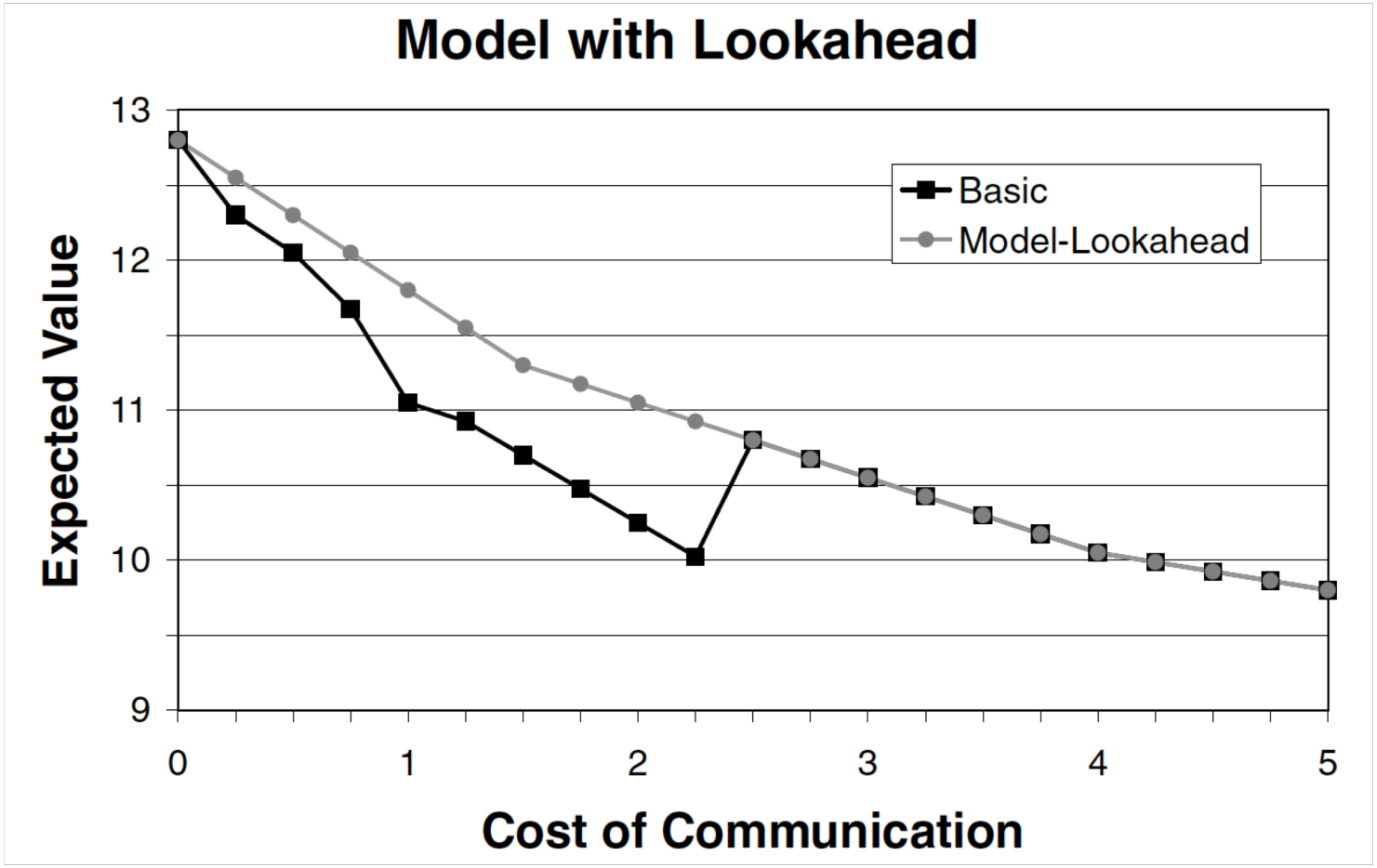
$$\mathrm{VoC}\left(s_i, \langle s_i^0, s_j^0 \rangle, t\right) = \sum_{s_j} P\left(s_j \mid s_j^0, t, \pi_j^0\right)\left[V^*(s_i, s_j) - C - V(s_i, s_j)\right], \qquad (2)$$

- The myopic assumption – that this is the last chance to communicate – can lead to over communication
- But limited lookahead – considering the option to delay communication by one step – helps resolve the problem

Myopic Communication Strategies

Becker et al., CI 2009

- Limited lookahead is sufficient to get rid of anomalies



Language Learning

Allen, Goldman & Zilberstein, IJCAI 2005

- Learning to assign the correct meaning to messages received from other agents
- Maintaining belief-states over a "translation"
- Approach converges on coordinated communication and action over time
- Also explored in Reinforcement Learning
  (e.g., "Learning to Communicate and Act in Cooperative Multiagent Systems using Hierarchical RL" [Ghavamzadeh & Mahadevan, AAMAS 04])
- More recently explored in the context of ad hoc teams (e.g., "Communicating with Unknown Teammates" [Barrett et al. ECAI 14])

More Challenging Questions

- How to go beyond the “sync” model (sharing observations)? What communication language to use?
- When reasoning about explicit communication, how to factor implicit communication (via observations)?
- What are the right communication models for systems involving many agents (broadcast, one-to-one)?
- What are good cost models for communication, and how to factor the cost of processing new information?

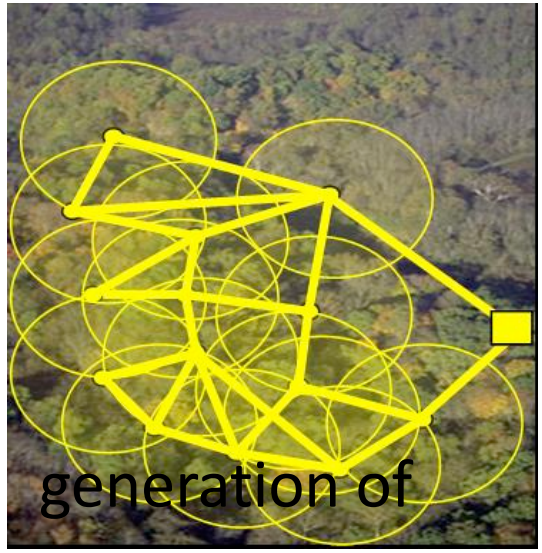
5. Organizational Design

- Organizational design is concerned with
  - Ways to assign local goals and tasks to agents and limit the scope of their decisions
  - Ways to limit the number of other agents that each agent must interact with
  - Ways to update the structure of the organization dynamically as the circumstances change
  - Ways to assign specific roles to agents in a team
  - Ways to decompose the overall objectives into local reward functions involving small groups of agents
- Examples
  - Sensor placement for detection or tracking
  - Seeding influential nodes in a social network
  - Interaction graph generation

Generation of Interaction Graphs

Yeoh, Kumar & Zilberstein, IJCAI 2013

- MA models such as ND-POMDP that involve ≫2 agents, are designed to exploit the locality of interaction
- Existing algorithms often assume a given, static interaction graph
- Initial work has automated the interaction graphs



- Greedy search via space of interaction graphs guided by exact value of best plan or heuristic estimates
- MILP can be used to find optimal design with respect to heuristic estimates.

More Challenging Questions

- How to generalize heuristic and exact methods for generating interaction graphs beyond ND-POMDPs
- How to design more complex organizations involving heterogeneous agents and dynamically changing IG
- How to optimize organizational design using an abstract model of the problem
- How to assign roles to agents to simplify coordination (e.g., assign an agent to each door of multiagent tiger)
- How to modify the environment to facilitate the development of an efficient organization

6. Social Norms and Commitments

- In social psychology: Social norms are "rules and standards that are understood by members of a group and that guide and/or constrain social behavior without the force of laws." [Cialdini Trost, 1998]
- Studied extensively in the social sciences, and also in AI

- Examples: Navigation through crowds while respecting traffic rules, or an EMS response protocol to avoid dispatching too many redundant services
- Crucial for effective agent and human-robot interaction

Prior Work on Social Norms

- "On the synthesis of useful social laws for artificial agent societies" [Shoham & Tennenholtz, AAAI 92]
- "Commitments and conventions: The foundation of coordination in multi-agent systems" [Jennings, 1993]
- "Emergent coordination through the use of cooperative state-changing rules" [Goldman & Rosenschein, 1993]
- "Optimal social laws" [Ågotnes & Wooldridge, AAMAS 2010]

- How can norms help solve DEC-POMDPs?
- How to best integrate norms with existing solvers?

Commitments

- Protocols for forming and managing commitments offer an important mechanism to simplify planning
- From a purely decision theoretic perspective commitments are questionable:
  - Why make any commitments that may limit the value of the policy?
  - How can agents make commitments when operating in a stochastic domain under limited observability?
- Bounded resources/rationality could justify commitments
- No significant work yet on commitments in DEC-POMDPs
- Some prior work by Durfee, Singh and others

Conclusions

- Identified six mechanisms to add structure to multiagent planning problems and potentially simplify solution
- It is crucial to allow algorithms to exploit such mechanisms in order to apply them in practice
- It is unrealistic to expect existing DEC-POMDP solvers to "discover" such mechanisms "on the fly"
- The utility of these mechanisms has been demonstrated (e.g. feature-based memory states & planning with options)
- Need to pursue such mechanisms even if they seem to compromise "optimality" or include "ad hoc" design choices
- Coming next, "How to Help our Planning Algorithms Produce Explainable Results?"

Questions?