Decentralized POMDPs:

A Framework for Multiagent Planning under Uncertainty

Frans Oliehoek
Outline

- Multiagent Systems & Uncertainty
- The Dec-POMDP model
- Policies and their values
  
- Planning for Dec-POMDPs
  - backward: DP
  - forward: heuristic search
Multiagent Systems (MASs)

Why MASs?

- 1 intelligent agents → soon there will be many...
- Physically distributed systems: centralized solutions expensive and brittle.
- Can potentially provide [Vlassis, 2007, Sycara, 1998]
  - Speedup and efficiency
  - Robustness and reliability (‘graceful degradation’)
  - Scalability and flexibility (adding additional agents)
Uncertainty

- Outcome Uncertainty

- Partial Observability

- Multiagent Systems: uncertainty about others
Single-Agent Decision Making

- Background: MDPs & POMDPs

- An MDP \( \langle S, A, P_T, R, h \rangle \)
  - \( S \) – set of states
  - \( A \) – set of actions
  - \( P_T \) – transition function
  - \( R \) – reward function
  - \( h \) – horizon (finite)

- A POMDP \( \langle S, A, P_T, O, P_O, R, h \rangle \)
  - \( O \) – set of observations
  - \( P_O \) – observation function
  - \( P(s' \mid s, a) \)
  - \( R(s, a) \)
  - \( P(o \mid a, s') \)
Example: Predator-Prey Domain

- Predator-Prey domain
  - 1 agent: predator
  - prey: part of environment
  - on a torus

- Formalization:
  - states
  - actions
  - transitions
  - rewards
Example: Predator-Prey Domain

- Predator-Prey domain
  - 1 agent: predator
  - prey: part of environment
  - on a torus

- Formalization:
  - states: (-3,4)
  - actions: N, W, S, E
  - transitions: failing to move, prey moves
  - rewards: reward for capturing
Example: Predator-Prey Domain

- Predator-Prey domain
  - 1 agent: predator
  - prey: part of environment on a torus

Formalization:
- states (-3,4)
- actions N,W,S,E
- transitions failing to move, prey moves
- rewards reward for capturing

Markov decision process (MDP)
- Markovian state $s$... (which is observed!)
- policy $\pi$ maps states $\rightarrow$ actions
- Value function $Q(s,a)$
- Compute via value iteration / policy iteration

\[
Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')
\]
- rewards
- reward for capturing

Markov decision process (MDP)
- Markovian state $s$... (which is observed!)
- policy $\pi$ maps states $\rightarrow$ actions
- Value function $Q(s,a)$
- Compute via value iteration / policy iteration

\[
Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')
\]
Partial Observability

- Now: partial observability
  - E.g., limited range of sight

- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)

\[ o = 'nothing' \]
Partial Observability

- Now: partial observability
  - E.g., limited range of sight

- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)

\[ o = (-1,1) \]
Partial Observability

- Now: partial observability
  - E.g., limited range of sight

- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)

Can not observe the state
→ Need to maintain a belief over states $b(s)$
→ Policy maps beliefs to actions $\pi(b) = a$
Partial Observability

- Now: partial observability
  - E.g., Partially Observable MDP (POMDP)
  - MDP + observations
    - explicit observations
    - observation probabilities
      - noisy observations (detection probability)

Can not observe the state
→ Need to maintain a belief over states $b(s)$
→ Policy maps beliefs to actions $\pi(b) = a$
Now: partial observability

- Partially Observable MDP (POMDP)
  - reduction → continuous state MDP (in which the belief is the state)
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)

Can not observe the state → Need to maintain a belief over states $b(s)$ → Policy maps beliefs to actions $\pi(b) = a$
Partial Observability

- Now: partial observability

  - Partially Observable MDP (POMDP)
    - reduction → continuous state MDP
      (in which the belief is the state)
    - Value iteration:
      - make use of α-vectors (↔ complete policies)
      - perform pruning

  - MDP with explicit observations
  - noisy observations (detection probability)

\[ V(b) \]

\[ s_1 \quad \text{belief} \quad s_2 \]
Now: multiple agents
  - fully observable

Formalization:
  - states
  - actions
  - joint actions
  - transitions
  - rewards
Multiple Agents

- Now: multiple agents
  - fully observable

- Formalization:
  - states: $((3,-4), (1,1), (-2,0))$
  - actions: $\{N,W,S,E\}$
  - joint actions: $\{(N,N,N), (N,N,W), \ldots, (E,E,E)\}$
  - transitions: probability of failing to move, prey moves
  - rewards: reward for capturing jointly
Multiple Agents

- Now: multiple agents
  - fully observable

Formalization:
- states: \((3,-4), (1,1), (-2,0)\)
- actions: \(\{N,W,S,E\}\)
- joint actions: \(\{(N,N,N), (N,N,W), \ldots, (E,E,E)\}\)
- transitions: probability of failing to move, prey moves
- rewards: reward for capturing jointly

Multiagent MDP [Boutilier 1996]

- Differences with MDP
  - \(n\) agents
  - joint actions \(a = \langle a_1, a_2, \ldots, a_n \rangle\)
  - transitions and rewards depend on joint actions

- Solution:
  - Treat as normal MDP with 1 'puppeteer agent'
  - Optimal policy \(\pi(s) = a\)
  - Every agent executes its part

\[a = \langle a_1, a_2, \ldots, a_n \rangle\]

\[\pi(s) = a\]
Multiple Agents

- Now: multiple agents
  - fully observable
  - formalization:
    - states: \((3,-4), (1,1), (-2,0)\)
    - actions: \(\{N,W,S,E\}\)
    - joint actions: \(\{(N,N,N), (N,N,W), \ldots, (E,E,E)\}\)
    - transitions: probability of failing to move, prey moves
    - rewards: reward for capturing jointly

Multiagent MDP [Boutilier 1996]

- Differences with MDP
  - \(n\) agents
  - joint actions \(a = \langle a_1, a_2, \ldots, a_n \rangle\)
  - transitions and rewards depend on joint actions

- Solution:
  - Treat as normal MDP with 1 'puppeteer agent'
  - Optimal policy \(\pi(s) = a\)
  - Every agent executes its part

- Rewards
  - reward for capturing jointly

Catch: …?
Multiple Agents

- Now: multiple agents
  - fully observable

- Formulation:
  - states \((3,-4), (1,1), (-2,0)\)
  - actions \(\{N, W, S, E\}\)
  - joint actions \(\{(N,N,N), (N,N,W), \ldots, (E,E,E)\}\)
  - transitions: probability of failing to move, prey moves
  - rewards: reward for capturing jointly

Catch: number of joint actions is exponential!
(but other than that, conceptually simple.)

- Differences with MDP
  - \(n\) agents
  - joint actions \(a = \langle a_1, a_2, \ldots, a_n \rangle\)
  - transitions and rewards depend on joint actions

- Solution:
  - Treat as normal MDP with 1 'puppeteer agent'
  - Optimal policy \(\pi(s) = a\)
  - Every agent executes its part

- rewards: reward for capturing jointly
Now both...

- partial observability
- multiple agents
Multiple Agents & Partial Observability

- Now both...
  - partial observability
  - multiple agents

- Decentralized POMDPs (Dec-POMDPs) [Bernstein et al. 2002]

- both
  - joint actions and
  - joint observations
Multiple Agents & Partial Observability

- Again we can make a reduction...

any idea?
Multiple Agents & Partial Observability

- Again we can make a reduction...
  Dec-POMDPs $\rightarrow$ MPOMDP
  (multiagent POMDP)

- 'puppeteer agent'
  - receives joint observations
  - takes joint actions

- requires broadcasting observations!
  - instantaneous, cost-free, noise-free communication $\rightarrow$ optimal
    [Pynadath and Tambe 2002]
  - Without such communication: no easy reduction.
The Dec-POMDP Model
Acting Based On Local Observations

- MPOMDP: Act on global information
- Can be impractical:
  - communication not possible
  - significant cost (e.g. battery power)
  - not instantaneous or noise free
  - scales poorly with number of agents!

- Alternative: act based only on local observations
  - Other side of the spectrum: no communication at all
  - (Also other intermediate approaches: delayed communication, stochastic delays)
Formal Model

- A Dec-POMDP
  - $\langle S, A, P_T, O, P_O, R, h \rangle$
  - $n$ agents
  - $S$ – set of states
  - $A$ – set of joint actions
  - $P_T$ – transition function
  - $O$ – set of joint observations
  - $P_O$ – observation function
  - $R$ – reward function
  - $h$ – horizon (finite)
Running Example

- 2 generals problem
Running Example

- 2 generals problem

\[
S = \{ s_L, s_S \} \\
A_i = \{ \text{(O)bserve, (A)ttack} \} \\
O_i = \{ \text{(L)arge, (S)mall} \}
\]

Transitions
- Both Observe $\rightarrow$ no state change
- At least 1 Attack $\rightarrow$ reset (50% probability $s_L, s_S$)

Observations
- Probability of correct observation: 0.85
- E.g., $P(<L, L> | s_L) = 0.85 \times 0.85 = 0.7225$
- (reset is not observed!)
Running Example

- **2 generals problem**

  \[ S = \{ s_L, s_S \} \]
  \[ A_i = \{ (O)bserve, (A)ttack \} \]
  \[ O_i = \{ (L)arge, (S)mall \} \]

  **Rewards**
  - 1 general attacks: he loses the battle
    - \( R(\ast, <A,O>) = -10 \)
  - Both generals Observe: small cost
    - \( R(\ast, <O,O>) = -1 \)
  - Both Attack: depends on state
    - \( R(s_L, <A,A>) = -20 \)
    - \( R(s_S, <A,A>) = +5 \)
Running Example

2 generals problem

\[ S = \{ s_L, s_S \} \]
\[ A_i = \{ \text{(O)bserve}, \text{(A)ttack} \} \]
\[ O_i = \{ \text{(L)arge}, \text{(S)mall} \} \]

Rewards

- 1 general attacks: he loses the battle
  - \( R(*, <A,O>) = -10 \)
- Both generals Observe: small cost
  - \( R(*, <O,O>) = -1 \)
- Both Attack: depends on state
  - \( R(s_L, <A,A>) = -20 \)
  - \( R(s_R, <A,A>) = +5 \)

Suppose \( h = 3 \), what do you think is optimal in this problem?
Related Frameworks

- Partially observable stochastic games [Hansen et al. 2004]
  - Non-identical payoff

- Interactive POMDPs [Gmytrasiewicz & Doshi 2005, JAIR]
  - Subjective view of MAS

- Imperfect information extensive form games
  - Represented by game tree
  - E.g., poker [Sandholm 2010, AI Magazine]

Rest of lecture: planning for Dec-POMDPs...
Off-line / On-line phases

- off-line planning, on-line execution is decentralized

\[ \pi = \langle \pi_1, \pi_2 \rangle \]

- (Smart generals make a plan in advance!)
Policies and their Values
Policy Domain

- What do policies look like?
  - In general histories → actions
  - in MDP/POMDP: more compact representations...
- Now, this is difficult: no such representation known!
  → So we will be stuck with histories
Policy Domain

What do policies look like?
- In general histories → actions
- in MDP/POMDP: more compact representations...

Now, this is difficult: no such representation known!
→ So we will be stuck with histories

Most general, AOHs:
\[(a_i^0, o_i^1, a_i^1, ..., a_i^{t-1}, o_i^t)\]

But: can restrict to deterministic policies
→ only need OHs:
\[\vec{o}_i = (o_i^1, ..., o_i^t)\]
No Compact Representation?

- **Joint Belief,** $b(s)$ (as in MPOMDP) [Pynadath and Tambe 2002]
  - compute $b(s)$ using joint actions and observations
  - Problem: ?
No Compact Representation?

- **Joint Belief**, $b(s)$ (as in MPOMDP) [Pynadath and Tambe 2002]
  - compute $b(s)$ using joint actions and observations
  - Problem: agents do not know those during execution
Goal of Planning

- Find the **optimal** joint policy $\pi^* = \langle \pi_1, \pi_2 \rangle$
  - where individual policies map OHs to actions $\pi_i : \tilde{O}_i \rightarrow A_i$

- What is the optimal one?
  - Define **value** as the expected sum of rewards:
    
    $$V(\pi) = E\left[ \sum_{t=0}^{h-1} R(s,a) \mid \pi, b^0 \right]$$

- optimal joint policy is one with maximal value (can be more that achieve this)
Goal of Planning

- **Find the optimal joint policy**
  - where individual policies map OHs to actions $\pi_i: D_i \rightarrow A_i$
  - Define value as the expected sum of rewards:
    $$V(\pi) = \mathbb{E}\left[\sum_{t=0}^{h-1} R(s, a) \mid \pi, b^0\right]$$
  - Optimal joint policy is one with maximal value
    $$\pi^* = \left\{ \pi_1, \pi_2 \right\}$$

Optimal policy for 2 generals, h=3

<table>
<thead>
<tr>
<th>OH State Sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>observe</td>
</tr>
<tr>
<td>(o_small)</td>
<td>observe</td>
</tr>
<tr>
<td>(o_large)</td>
<td>observe</td>
</tr>
<tr>
<td>(o_small, o_small)</td>
<td>attack</td>
</tr>
<tr>
<td>(o_small, o_large)</td>
<td>attack</td>
</tr>
<tr>
<td>(o_large, o_small)</td>
<td>attack</td>
</tr>
<tr>
<td>(o_large, o_large)</td>
<td>observe</td>
</tr>
</tbody>
</table>

Value = -2.86743
Goal of Planning

- Find the optimal joint policy
  - where individual policies map actions

What is the optimal one?

Define value as the expected sum of rewards:

$$\pi^* = \langle \pi_1, \pi_2 \rangle$$

$$\pi_i: \vec{O}_i \rightarrow A_i$$

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

Optimal policy for 2 generals, $h=3$

- value=-2.86743
  - () --> observe
  - (o_small) --> observe
  - (o_large) --> observe
  - (o_small, o_small) --> attack
  - (o_small, o_large) --> attack
  - (o_large, o_small) --> attack
  - (o_large, o_large) --> observe

Conceptually:

what should policy optimize to allow for good coordination (thus high value)?
Coordination vs. Exploitation of Local Information

- Inherent trade-off

  **coordination vs. exploitation of local information**

  - Ignore own observations → 'open loop plan'
    - E.g., “ATTACK on 2nd time step”
      + maximally predictable
      - low quality
  
  - Ignore coordination → 'MPOMDP plan'
    - E.g., 'individual belief' \( b_i(s) \) and execute the MPOMDP policy
      + uses local information
      - likely to result in mis-coordination

- Optimal policy \( \pi^* \) should balance between these!
Value of a Joint Policy

- Sub-tree policies:

- Given a particular joint policy $\pi = q^{\tau=h}$
  $\rightarrow$ Just a (complex) Markov Chain

- Value:

$$V(\tilde{\theta}, q^{\tau=k}) = R(\tilde{\theta}, a) + \sum_o P(o|\tilde{\theta}, a) V(\tilde{\theta}', q^{\tau=k-1})$$
Optimal Value Functions – 1

- Optimal value functions are difficult!
- Consider selecting the best joint sub-tree policy $q^T$

- We *can* compute value... ...but *cannot* select the maximizing $q^T$ independently!
**Optimal Value Functions – 2**

- *Cannot* select the maximizing \(q^t\) independently...
  - Need to reason over assignment for all AOHs of a stage \(t\) simultaneously!

- Value stage \(t\)
  \[
  \sum_\theta P(\theta|b^0, \phi) V(\theta, q^{\tau=h-t}) = \sum_{\langle \theta_1, \theta_2 \rangle} P(\langle \theta_1, \theta_2 \rangle|b^0, \phi) V(\langle \theta_1, \theta_2 \rangle, \langle q_1, q_2 \rangle)
  \]

- Find mappings \(\Gamma_1, \Gamma_2\) (from AOHs → sub-tree policies) that maximize
  \[
  \sum_{\langle \theta_1, \theta_2 \rangle} P(\langle \theta_1, \theta_2 \rangle|b^0, \phi) V(\langle \theta_1, \theta_2 \rangle, \langle \Gamma_1(\theta_1), \Gamma_2(\theta_2) \rangle)
  \]

  dependence on history  
  dependence on future
Planning Methods
Brute Force Search

- We can compute the value of a joint policy $V(\pi)$

- So the **stupidest algorithm** is:
  - compute $V(\pi)$, for all $\pi$
  - select a $\pi$ with maximum value

- Number of joint policies is huge! (doubly exponential in horizon $h$)
- Clearly intractable...

<table>
<thead>
<tr>
<th>$h$</th>
<th>num. joint policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>16384</td>
</tr>
<tr>
<td>4</td>
<td>1.0737e+09</td>
</tr>
<tr>
<td>5</td>
<td>4.6117e+18</td>
</tr>
<tr>
<td>6</td>
<td>8.5071e+37</td>
</tr>
<tr>
<td>7</td>
<td>2.8948e+76</td>
</tr>
<tr>
<td>8</td>
<td>3.3520e+153</td>
</tr>
</tbody>
</table>
Brute Force Search

- We can compute the value of a joint policy $V(\pi)$

So the stupidest algorithm is:
- compute $V(\pi)$, for all $\pi$

No easy way out...

The problem is **NEXP-complete** [Bernstein et al. 2002]

- The number of joint policies is huge!
  - $(\text{doubly exponential in horizon } h)$

- Clearly intractable...

<table>
<thead>
<tr>
<th>$h$</th>
<th>num. joint policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>16384</td>
</tr>
<tr>
<td>4</td>
<td>$1.0737e+09$</td>
</tr>
<tr>
<td>5</td>
<td>$4.6117e+18$</td>
</tr>
<tr>
<td>6</td>
<td>$8.5071e+37$</td>
</tr>
<tr>
<td>7</td>
<td>$2.8948e+76$</td>
</tr>
<tr>
<td>8</td>
<td>$3.3520e+153$</td>
</tr>
</tbody>
</table>
Brute Force Search

- We can compute the value of a joint policy $V(\pi)$

So the stupidest algorithm is:

- compute $V(\pi)$ for all $\pi$

The problem is **NEXP-complete** [Bernstein et al. 2002]

most likely (assuming EXP $\neq$ NEXP)

doubly exponential time required.

<table>
<thead>
<tr>
<th>$h$</th>
<th>num. joint policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>16384</td>
</tr>
<tr>
<td>4</td>
<td>1.0737e+09</td>
</tr>
<tr>
<td>5</td>
<td>4.6171e+18</td>
</tr>
<tr>
<td>6</td>
<td>8.5071e+37</td>
</tr>
<tr>
<td>7</td>
<td>2.8948e+76</td>
</tr>
<tr>
<td>8</td>
<td>2.3521e+153</td>
</tr>
</tbody>
</table>

Clearly:

- Still, there are better algorithms that work better for at least some problems...

- Useful to gain understanding about problem.
Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.

![Diagram](image-url)
Generate all policies in a special way:

- from 1 stage-to-go policies $Q^{t=1}$

Exhaustive backup operation.
Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{t=1}$
- Exhaustive backup operation

$Q^t_i$
Generate all policies in a special way:

- from 1 stage-to-go policies $Q^{\tau=1}$
- construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.

Exhaustive backup operation
Generate all policies in a special way:

- from 1 stage-to-go policies $Q^{t=1}$

Exhaustive backup operation
Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{t=1}$
  - construct all 2-stages-to-go policies $Q^{t=2}$, etc.

Exhaustive backup operation

a new $q^{t+1}$
Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{t=1}$

Exhaustive backup operation

To generate all $Q^{t+1}$
- All actions
- All assignments of $q^t$ to observations
(obviously) this scales very poorly...

\[ Q_1^{\tau=1} \]

\[ Q_2^{\tau=1} \]
(obviously) this scales very poorly...

\[ Q_1^{\tau=2} \]

\[ Q_2^{\tau=2} \]
Dynamic Programming – 2

- (obviously) this scales very poorly...

\[ Q_1^{\tau=3} \]

\[ Q_2^{\tau=3} \]
(obviously) this scales very poorly...

This does not get us anywhere!

but...

<table>
<thead>
<tr>
<th>h</th>
<th>num. indiv. policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>32768</td>
</tr>
<tr>
<td>5</td>
<td>2.1475e+09</td>
</tr>
<tr>
<td>6</td>
<td>9.2234e+18</td>
</tr>
<tr>
<td>7</td>
<td>1.7014e+38</td>
</tr>
<tr>
<td>8</td>
<td>5.7896e+76</td>
</tr>
</tbody>
</table>
Dynamic Programming – 3

- Perhaps not all those \( Q_i^\tau \) are useful!
  - Perform **pruning** of 'dominated policies'!

- Algorithm [Hansen et al. 2004]

\[
Q_i^{\tau=1} = A_i
\]

Initialize \( Q1(1), Q2(1) \)

for \( \tau = 2 \) to \( h \)

\[
\begin{align*}
Q1(\tau) &= \text{ExhaustiveBackup}(Q1(\tau-1)) \\
Q2(\tau) &= \text{ExhaustiveBackup}(Q2(\tau-1)) \\
\text{Prune}(Q1, Q2, \tau)
\end{align*}
\]

end
Perhaps not all those $Q_i^\tau$ are useful!

- Perform **pruning** of 'dominated policies'!

**Algorithm** [Hansen et al. 2004]

```
Q_{i=1}^\tau = A_i
```

Initialise $Q_1(1), Q_2(1)$

for $\tau = 2$ to $h$

$Q_1(\tau) = \text{ExhaustiveBackup}(Q_1(\tau-1))$

$Q_2(\tau) = \text{ExhaustiveBackup}(Q_2(\tau-1))$

Prune($Q_1, Q_2, \tau$)

end

Note: cannot prune independently!

- usefulness of a $q_1$ depends on $Q_2$
- and vice versa
  → **Iterated elimination** of policies
Perhaps not all those $Q_i^\tau$ are useful!
- Perform pruning of 'dominated policies'!

Algorithm [Hansen et al. 2004]

```
Initialize Q1(1), Q2(1)
for tau=2 to h
    Q1(tau) = ExhaustiveBackup(Q1(tau-1))
    Q2(tau) = ExhaustiveBackup(Q2(tau-1))
    Prune(Q1,Q2,tau)
end
```

Note: cannot prune independently!
- usefulness of a $q_1$ depends on $Q_2$
- and vice versa
  → Iterated elimination of policies

pruning itself: via LP [Hansen et al. 2000]
Dynamic Programming – 4

- Initialization

\[ Q_1^{\tau=1} \]

\[ Q_2^{\tau=1} \]
Exhaustive Backups gives

\[ Q^{\tau=2}_1 \]

\[ Q^{\tau=2}_2 \]
Dynamic Programming – 4

- Pruning agent 1...

\[ Q_{1}^{\tau=2} \]

\[ Q_{2}^{\tau=2} \]

Hypothetical Pruning
(not the result of actual pruning)
Dynamic Programming – 4

- Pruning agent 2...

\[ Q_{1}^{\tau=2} \]

\[ Q_{2}^{\tau=2} \]
Pruning agent 1...

\[ Q_{1}^{\tau=2} \]

\[ Q_{2}^{\tau=2} \]
Etc...

\[ Q_1^{\tau=2} \]

\[ Q_2^{\tau=2} \]
Dynamic Programming – 4

- Etc...

\[ Q_1^{\tau=2} \]

\[ Q_2^{\tau=2} \]

In this case: symmetric → but need not be in general!
Exhaustive backups:

- $Q^{\tau=3}_1$
- $Q^{\tau=3}_2$

We avoid generation of many policies!
Exhaustive backups:

\[ Q_1^{\tau=3} \]

\[ Q_2^{\tau=3} \]
Pruning agent 1...

\[ Q_1^{\tau=3} \]

\[ Q_2^{\tau=3} \]
Pruning agent 2...

\[ Q_{\tau=3}^1 \]

\[ Q_{\tau=3}^2 \]
Dynamic Programming – 4

- Etc...

\[ Q_1^{\tau=3} \]

\[ Q_2^{\tau=3} \]
Dynamic Programming – 4

- Etc...

At the very end:

\[ Q^{t=3} \]

\[ \ldots ? \]
At the very end:

- evaluate all the remaining combinations of policies
- select the best one

\[ V(q^{\tau=h}) = \sum_s b_0(s)V(s, q^{\tau=h}) \]
Bottom-up vs. Top-down

- **DP constructs bottom-up**
- **Alternatively try and construct top down**
  
  $\rightarrow$ leads to (heuristic) search [Szer et al. 2005, Oliehoek et al. 2008]
Heuristic Search – Intro

- Core idea is the same as DP:
  - incrementally construct all (joint) policies
  - try to avoid work

- Differences
  - different starting point and increments
  - use **heuristics** (rather than pruning) to avoid work
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'

![Diagram showing a tree structure with 'O', 'A', and 'S' nodes, and 'L' edges, illustrating a joint policy tree.]
Heuristic Search – 1

- Incrementally construct all (joint) policies
- 'forward in time'

Start with unspecified policy

1 partial joint policy

Start with unspecified policy
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'

1 partial joint policy
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'

1 partial joint policy
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'

1 complete joint policy (full-length)
Creating **ALL** joint policies → tree structure!

Root node: unspecified joint policy
Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

Creating a child node: assignment actions at $t=0$
Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

Node expansion: create all children
Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

$t=0$
Heuristic Search – 2

- Creating **ALL** joint policies $\rightarrow$ tree structure!

Expand next node...
How many children?

$t = 1$

...
Creating **ALL** joint policies → tree structure!

Many more children!

need to assign action to 4 OHs now: $2^4 = 16$

$t=1$

...
Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

\[ t = 2 \]

Last stage: even more!

need to assign action to 8 OHs now: \(2^8 = 256\) children
(for each node at level 2!)
Heuristic Search – 3

- too big to create completely...
- Idea: use **heuristics**
  - avoid going down non-promising branches!
- Apply A* → **Multiagent A*** [Szer et al. 2005]
Heuristic Search – 3

- too big to create completely
  - Idea: use heuristics
    - avoid going down non-promising branches!
  - Apply A* → Multiagent A*
    - [Szer et al. 2005]

Main intuition A*

- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]
Heuristic Search – 3

- too big to create completely
- Idea: use heuristics to avoid going down non-promising branches!
- Apply A* → Multiagent A* [Szer et al. 2005]

Main intuition A*
- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]
### Heuristic Search – 3

- too big to create completely
- Idea: use heuristics to avoid going down non-promising branches!
- Apply A* → Multiagent A* [Szer et al. 2005]

#### Main intuition A*
- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]

Select highest valued node & expand...

- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]
**Heuristic Search – 3**

- too big to create completely
- Idea: use heuristics to avoid going down non-promising branches!
- Apply A* → Multiagent A* [Szer et al. 2005]

**Main intuition A***

- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]
Heuristic Search – 3

- too big to create completely...
- Idea: use heuristics to avoid going down non-promising branches!
- Apply A* → Multiagent A*
  [Szer et al. 2005]

Main intuition A*:
- For each node, compute F-value
- Select next node based on F-value
- More info: [Russel&Norvig 2003]

F-Value of a node n:
- F(n) is a optimistic estimate
- i.e., F(n) >= V(n') for any descendant n' of n
- F(n) = G(n) + H(n)

- reward up to n (for first t stages)
- Optimistic estimate of reward below n (reward for stages t,t+1,...,h-1)

For each node, compute F-value
Select next node based on F-value
More info: [Russel&Norvig 2003]
Heuristic Search – 4

- Use heuristics $F(n) = G(n) + H(n)$

- $G(n)$ – actual reward of reaching $n$
  - a node at depth $t$ specifies $\phi^t$ (i.e., actions for first $t$ stages)
  - $\rightarrow$ can compute $V(\phi^t)$ over stages $0...t-1$

- $H(n)$ – should overestimate!
  - pretend that it is an MDP, or POMDP: $\hat{Q}_{MDP}, \hat{Q}_{POMDP}$
  - compute

\[
H(n) = H(\phi^t) = \sum_s P(s|\phi^t, b^0) \hat{Q}(s)
\]
Further Developments

- **DP**
  - Improvements to exhaustive backup [Amato et al. 2009]
  - Compression of values (LPC) [Boularias & Chaib-draa 2008]
  - (Point-based) Memory bounded DP [Seuken & Zilberstein 2007a]
  - Improvements to PB backup [Seuken & Zilberstein 2007b, Carlin and Zilberstein, 2008; Dibangoye et al, 2009; Amato et al, 2009; Wu et al, 2010, etc.]

- **Heuristic Search**
  - No backtracking: just most promising path [Emery-Montemerlo et al. 2004, Oliehoek et al. 2008]
  - Clustering of histories: reduce number of child nodes [Oliehoek et al. 2009]
  - Incremental expansion: avoid expanding all child nodes [Spaan et al. 2011]
  - **MILP** [Aras and Dutech 2010]
State of The Art

To get an impression...

- **Optimal solutions**
  - Improvements of MAA* lead to significant increases
  - but problem dependent

- **Approximate (no quality guarantees)**
  - MBDP: linear in horizon [Seuken & zilberstein 2007a]
  - Rollout sampling extension: up to 20 agents [Wu et al. 2010b]
  - Transfer planning: use smaller problems to solve large (structured) problems (up to 1000) agents [Oliehoek et al. 2013]
Further Topics

- Infinite-horizon planning
- Communication:
  - implicit/explicit
  - delays
  - costs
- Structured Models
  - e.g., factored Dec-POMDPs
- Reinforcement learning
References

- References can be found in

Some Further Topics
Further topics
- Communication
- Infinite Horizon
- Reinforcement Learning
Communication

- instantaneous, cost-free, and noise-free:
  - Dec-MDP → multiagent MDP (MMDP)
  - Dec-POMDP → multiagent POMDP (MPOMDP)

- but in practice:
  - probability of failure
  - delays
  - costs

- Also: implicit communication! (via observations and actions)
Implicit Communication

- Encode communications by actions and observations

- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]
Implicit Communication

- Encode communications by actions and observations

- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]
Implicit Communication

- Encode communications by actions and observations

- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]

- E.g. communication bit
  - doubles the #actions and observations!
  - Clearly, useful... but intractable for general settings (perhaps for analysis of very small communication systems)
Explicit Communication

- perform a particular information update (e.g., sync) as in the MPOMDP:
  - each agent broadcasts its information, and
  - each agent uses that to perform joint belief update

- Other approaches:
  - Communication cost [Becker et al. 2005]
  - Delayed communication [Hsu et al. 1982, Spaan et al. 2008, Oliehoek & Spaan 2012]
  - Communicate every k stages [Goldman & Zilberstein 2008]
Infinite-horizon Dec-POMDPs

- Infinite-horizon case: undecidable.
- Can compute $\varepsilon$-approximate solution

- Use finite-state controllers to represent policies.
  - 'back up' operations on controllers, [Bernstein et al. 2009]
  - BPI [Bernstein et al, 2005].
  - NLP [Amato et al, 2010].
Reinforcement Learning

- All this assumed the model is given, if not the case: not a great deal of work
  - Plenty of MARL [Busoniu et al, 2008] but not for the general Dec-POMDP setting...

- Exceptions:
  - decentralized gradient ascent [Peshkin et al, 2000]
  - single-agent methods (e.g., Q-learning) [Claus and Boutilier 1998, Crites and Barto 1998]
  - Centralized sample-based planning [Wu et al 2010b]

- problems:
  - when/how the agents observe the rewards? (episodes?)
  - how to learn about coupled dynamics from only individual observations? (cannot even compute a belief with the model!)
  - learning in a POMDP is hard!
Extra Slides...
No Compact Representation?

There are a number of types of beliefs considered

- **Joint Belief**, $b(s)$ (as in MPOMDP) [Pynadath and Tambe 2002]
  - compute $b(s)$ using joint actions and observations
  - Problem: agents do not know those during execution

- **Multiagent belief**, $b_i(s, q_{-i})$ [Hansen et al. 2004]
  - Belief over future policies of other agents, $q_{-i}$
  - Need to be able to predict the other agents!
    - for belief update $P(s'|s, a_i, a_{-i})$, $P(o|a_i, a_{-i}, s')$, and prediction of $R(s, a_i, a_{-i})$
  - form of those other policies?
    - most general: $\pi_j: \vec{o}_j \rightarrow a_j$
    - if they use beliefs? $\rightarrow$ infinite recursion of beliefs!
Coordination vs. Exploitation of Local Information

- Inherent trade-off

**coordination vs. exploitation of local information**

- Ignore own observations → 'open loop plan'
  - E.g., “ATTACK on 2nd time step”
    + maximally predictable
    - low quality
- Ignore coordination
  - E.g., 'individual belief' $b_i(s)$ and execute the MPOMDP policy
    + uses local information
    - likely to result in mis-coordination

- Optimal policy $\pi^*$ should balance between these!
Value of a Joint Policy

- Sub-tree policies:

\[ V(s, q^{\tau=k}) = R(s, a) + \sum_{s'} \sum_{o} P(s'|o|s,a) V(s', q^{\tau=k-1}) \]

- Given a particular joint policy \( \pi = q^{\tau=h} \)
  - \( \rightarrow \) Just a (complex) Markov Chain
  - Augmented state \( \langle s, q^{\tau=k} \rangle \)
Optimal Value Functions – 1

- Optimal value functions are difficult!

- consider selecting the best joint sub-tree policy $q^\tau$

- We can compute value
  
  $$V(\theta, q^{\tau=k}) = \sum_s P(s|\theta, b^0) V(s, q^{\tau=k})$$

- but cannot select the maximizing $q^\tau$ independently!