Decision making under uncertainty

Matthijs Spaan\textsuperscript{1} and Frans Oliehoek\textsuperscript{2}

\textsuperscript{1} Delft University of Technology
\textsuperscript{2} Maastricht University

Part 3: Multiagent Frameworks

14th European Agent Systems Summer School (EASSS '12)
Valencia, Spain

www.st.ewi.tudelft.nl/~mtjspaan/tutorialDMuU/
Multiagent Systems (MASs)

Why MASs?

- If we can make 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)
Example: Predator-Prey Domain

Predator-Prey domain – still single agent!

1 agent: the predator (blue)
prey (red) is part of the environment
on a torus ('wrap around world')

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

- Predator-Prey domain
  - 1 agent: the predator (blue)
  - prey (red) is part of the environment
  - on a torus ('wrap around world')

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

- Predator-Prey domain
  - 1 agent: the predator (blue), prey (red) is part of the environment
  - on a torus ('wrap around world')

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

- Predator-Prey domain
  - 1 agent: the predator (blue)
  - prey (red) is part of the environment
    - on a torus ('wrap around world')
  - Formalization:
    - states (-3,4)
    - actions N,W,S,E
    - transitions probability of 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 \( Q(s,a) \)
- Value iteration: way to compute it.
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 $\pi(b) = a$
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 $\pi(b) = a$

  Partially Observable MDP (POMDP)
  - reduction → continuous state MDP
    (in which the belief is the state)
  - Value iterations:
    - make use of $\alpha$-vectors
      (correspond to complete policies)
    - perform pruning: eliminate dominated $\alpha$'s

  Partially Observable MDP (POMDP)
  - reduction → continuous state MDP
    (in which the belief is the state)
  - Value iterations:
    - make use of $\alpha$-vectors
      (correspond to complete policies)
    - perform pruning: eliminate dominated $\alpha$'s

  $o = (-1,1)$
Multiple Agents

- 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

- 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

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

- 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: number of joint actions is exponential! (but other than that, conceptually simple.)
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?
Again we can make a reduction...

Dec-POMDPs $\rightarrow$ MPOMDP (multiagent POMDP)

- 'puppeteer' agent that
  - 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)

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

\[
P(s'|s, a)
\]

\[
o = \langle o_1, o_2, \ldots, o_n \rangle
\]

\[
P(o|a, s')
\]

\[
R(s, a)
\]
Running Example

- 2 generals problem

small army

large army
Running Example

- 2 generals problem

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

Transitions
- Both Observe: no state change
- At least 1 Attack: reset with 50% probability

Observations
- Probability of correct observation: 0.85
- E.g., \( P(<L, L> | s_L) = 0.85 \times 0.85 = 0.7225 \)
Running Example

- 2 generals problem

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

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
  \[ R(s_L,<A,A>) = -20 \]
  \[ R(s_R,<A,A>) = +5 \]
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
  - \( R(*,<A,O>) = -10 \)
- 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?
Off-line / On-line phases

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

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

- What do policies look like?
  - In general histories $\rightarrow$ actions
  - before: more compact representations...
  - Now, this is difficult: no such representation known!
    $\rightarrow$ So we will be stuck with histories
Policy Domain

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

Most general, AOHs:
\[
\phi_i = (a_{i0}^0, o_{i1}^1, a_{i1}^1, \ldots, a_{it-1}, o_{it})
\]

But: can restrict to deterministic policies
→ only need OHs:
\[
\tilde{\phi}_i = (o_{i1}^1, \ldots, o_{it})
\]
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:
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
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
  - 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: $\pi_j : \tilde{o}_j \rightarrow a_j$
    - if they use beliefs? → infinite recursion of beliefs!
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 $\pi^*$ where individual policies map OHs to actions $\pi_i: \tilde{O}_i \rightarrow A_i$

  where

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

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

  value $=-2.86743$

- What is the optimal one?

  Optimal policy for 2 generals, $h=3$

  \[
  \begin{array}{ccc}
  () & \rightarrow & \text{observe} \\
  (\text{o\_small}) & \rightarrow & \text{observe} \\
  (\text{o\_large}) & \rightarrow & \text{observe} \\
  (\text{o\_small, o\_small}) & \rightarrow & \text{attack} \\
  (\text{o\_small, o\_large}) & \rightarrow & \text{attack} \\
  (\text{o\_large, o\_small}) & \rightarrow & \text{attack} \\
  (\text{o\_large, o\_large}) & \rightarrow & \text{observe} \\
  \end{array}
  \]

- Define value as the expected sum of rewards:

  \[
  V(\pi) = E \left[ \sum_{t=0}^{h-1} R(s_t, a_t) \mid \pi, b^0 \right]
  \]

  where $R(s_t, a_t)$ is the reward at time $t$.

  \[
  \begin{array}{ccc}
  () & \rightarrow & \text{observe} \\
  (\text{o\_small}) & \rightarrow & \text{observe} \\
  (\text{o\_large}) & \rightarrow & \text{observe} \\
  (\text{o\_small, o\_small}) & \rightarrow & \text{attack} \\
  (\text{o\_small, o\_large}) & \rightarrow & \text{attack} \\
  (\text{o\_large, o\_small}) & \rightarrow & \text{attack} \\
  (\text{o\_large, o\_large}) & \rightarrow & \text{observe} \\
  \end{array}
  \]

  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
Goal of Planning

- **Find the optimal joint policy**
  - where individual policies map OHs to actions
  - What is the optimal one?
- **Define value** as the expected sum of rewards:
  \[ V(\pi) = E \left[ \sum_{t=0}^{h-1} R(s, a) \mid \pi, b^0 \right] \]
  - What should policy optimize to allow for good coordination (thus high value)?

**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

- () --> 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
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., compute an 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.
Planning Methods
Brute Force Search

- We can compute the value of a joint policy $V(\pi)$ using a Bellman-like equation [Oliehoek 2012]
- 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 $V(\pi)$
  - using a Bellman-like equation [Oliehoek 2012]

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

No easy way out...

The problem is \textbf{NEXP-complete} [Bernstein et al. 2002]

most likely (assuming EXP $\neq$ NEXP) doubly exponential time required.

(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 $V(\pi)$
  - using a Bellman-like equation [Oliehoek 2012]

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

No easy way out...

The problem is **NEXP-complete** [Bernstein et al. 2002] most likely (assuming EXP != 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>4.6117e+18</td>
</tr>
<tr>
<td>6</td>
<td>8.5071e+37</td>
</tr>
<tr>
<td>7</td>
<td>2.8948e+76</td>
</tr>
</tbody>
</table>

Clearly intractable

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

- Useful to understand what optimal really means! (trying to compute it helps understanding)
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.
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
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

Exhaustive backup operation

Exhaustive backup operation

Exhaustive backup operation

Exhaustive backup operation

Exhaustive backup operation
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
Dynamic Programming – 1

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

Exhaustive backup operation

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

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

\[ A \quad O \]

\[ A \quad O \]
(obviously) this scales very poorly...

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

\[ Q_2^{\tau=2} \]
(obviously) this scales very poorly...
(obviously) this scales very poorly...

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

\[ Q_2^{\tau=3} \]

<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>

This does not get us anywhere!

but...
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
Dynamic Programming – 3

- Perhaps not all those $Q_i^t$ are useful!
  - Perform **pruning** of 'dominated policies'!

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

  $Q_i^{t=1} = A_i$

  Initialize $Q_1(1)$, $Q_2(1)$
  
  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
Dynamic Programming – 4

- Initialization

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

\[ Q_2^{\tau=1} \]
Dynamic Programming – 4

- Exhaustive Backups gives

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

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

- Pruning agent 1...

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

\[ Q^{\tau=2}_2 \]

Hypothetical Pruning (not the result of actual pruning)
Pruning agent 2...

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

\[ Q_2^{\tau=2} \]
Pruning agent 1...

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

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

- Etc...

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

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

- Etc...

In this case: symmetric → but need not be in general!
Dynamic Programming – 4

- Exhaustive backups:

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

  \[ Q_2^{\tau=3} \]

  We **avoid** generation of many policies!
Dynamic Programming – 4

- Exhaustive backups:

\[ Q_1^{\tau=3} \quad Q_2^{\tau=3} \]
Pruning agent 1...

$Q_1^{\tau=3}$

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

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

\[ Q_{2}^{\tau=3} \]
Etc...

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

\[ Q_2^{\tau=3} \]
Etc...

At the very end:

\[ Q_1^{t=3} \]

\[ \ldots? \]

\[ Q_2^{t=3} \]
At the very end:

- evaluate all the remaining combinations of policies (i.e., the 'induced joint policies')
- select the best one
Bottom-up vs. Top-down

- DP constructs bottom-up
- Alternatively try and construct top down

→ 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'

---

1 joint policy
Heuristic Search – 1

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

Start with unspecified policy
Heuristic Search – 1

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

```
1 partial joint policy
```

```
O  S  L
|
S  ?  L  S  ?  L
|
S  ?  L  S  ?  L
```

```
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

why?
Creating **ALL** joint policies $\rightarrow$ 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!

\[ t = 0 \]

Diagram showing the creation of all joint policies at time 0, leading to a tree structure.
Creating **ALL** joint policies → tree structure!
- Creating **ALL** joint policies → tree structure!

\[ t = 1 \]

Many more children!

need to assign action to 4 OHs now: \( 2^4 = 16 \)
Heuristic Search – 2

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

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
- 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*

Main intuition A*
- For each node, compute F-value
- Select next node based on F-value

More info: [Russell & Norvig 2003]
Heuristic Search – 3

- too big to create complete search space
- 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)

• For each node, compute F-value
• Select next node based on F-value
• More info: [Russel&Norvig 2003]
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]
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 2010]
Related Areas

- 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]
Decision making under uncertainty

Matthijs Spaan\textsuperscript{1} and Frans Oliehoek\textsuperscript{2}

\textsuperscript{1} Delft University of Technology
\textsuperscript{2} Maastricht University

Part 4: Selected Further Topics

14th European Agent Systems Summer School (EASSS '12)
Valencia, Spain

www.st.ewi.tudelft.nl/~mtjspaan/tutorialDMuU/
Some Further Topics

Overview:

- On-line planning
- Communication
- Factored Models
  - Single Agent
  - Multiple agents

- Goal: present an overview of some high-level ideas
On-line Planning

- So far: planning in a separate off-line phase
- However: could also consider performing the planning during execution!
  - do not plan over entire space, but only those reachable in the (near) future!
  - but: need to plan at every step.
- In control theory 'receding horizon control' or 'model predictive control' (but details different)
Lookahead Planning

- Main idea: plan ahead for T stages
- Construct a tree of all possibilities and perform dynamic programming over this tree
Lookahead Planning

- Main idea: plan ahead for T stages
- Construct a tree of all possibilities and perform dynamic programming over this tree

```
S^0 ← now
S^1 -> S^1'
...  
S^2 -> S^2'
...  
S^3 -> S^3'
...  
```

This focuses computation on states that are reachable (in the near-future)
Lookahead Planning

- Main idea: plan ahead for T stages
- Construct a tree of all possibilities and perform dynamic programming over the tree
Lookahead Planning

- Main idea: plan ahead for T stages
- Construct a tree of all possibilities and perform dynamic programming over this tree

Expanding all possible next states → tree is huge...

- one idea: Sample!
- That works pretty good: bound independent of number of states [Kearns et al. 2002 ML]
Lookahead Planning

- Main idea: plan ahead for T stages
- Construct a tree of all possibilities and perform dynamic programming over the tree

Expanding all possible next states → tree is huge...

- One idea: Sample!
- That works pretty good: bound independent of number of states [Kearns et al. 2002 ML]

Still very big...

- Further idea: avoid expanding non-promising branches.
- Use upper confidence bounds
- UCT [Kocsis & Szepesvári, 2006 ECML]
Some Further Topics

Overview:

- On-line planning
- Communication
- Factored Models
  - Single Agent
  - Multiple agents
Communication

- Already discussed:
  - instantaneous cost-free and noise-free communication
    - Dec-MDP $\rightarrow$ multiagent MDP (MMDP)
    - Dec-POMDP $\rightarrow$ 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 \cite{Goldman2003, Spaan2006}
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 1982, Spaan 2008, Oliehoek 2012]
  - communicate every k stages [Goldman & Zilberstein 2008]
Some Further Topics

Overview:

- On-line planning
- Communication
- **Factored Models**
  - Single Agent
  - Multiple agents
Factored MDPs

- So far: used 'states'
- But in many problems states are **factored**
  - state is an assignment of variables $s=\langle f_1, f_2, \ldots, f_k \rangle$
  - factored MDP [Boutilier et al. 99 JAIR]

Examples:
- Predator-prey: x, y coordinate!
- Robotic P.A.
  - location of robot (lab, hallway, kitchen, mail room), tidiness of lab, coffee request, robot holds coffee, mail present, robot holds mail, etc.
  - Actions: move (2 directions), pickup coffee/mail, deliver coffee/mail
Factored States & Transitions

- loc
- tidy
- CR
- RHC
- M
- RHM
Factored States & Transitions

\[ S^t \]

- loc
- tidy
- CR
- RHC
- M
- RHM
Factored States & Transitions

$S^t$

$S^{t+1}$

loc
tidy
cr
rhc
m
rhm

loc
tidy
cr
rhc
m
rhm
Factored States & Transitions

$S^t$

loc
tidy
CR
RHC
M
RHM

$S^{t+1}$

loc
tidy
CR
RHC
M
RHM

“Move to lab”

$P(s^{t+1}|s^t, a = MTL)$

$P(s^{t+1}|s^t, a = MTL)$
Factored States & Transitions

\[ P(s^{t+1} | s^t, a = MTL) \]

But there is conditional independence!

E.g.: 'M' does not influence 'loc'
Factored States & Transitions

\[ S^t \rightarrow \text{"Move to lab"} \rightarrow S^{t+1} \]

- loc
- tidy
- CR
- RHC
- M
- RHM

"Move to lab"
Factored States & Transitions

$S_t$  $S_{t+1}$

- **loc** → **loc**
- **tidy** → **tidy**
- **CR** → **CR**
- **RHC** → **RHC**
- **M** → **M**
- **RHM** → **RHM**

"Move to lab"

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>H</th>
<th>K</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>1</td>
<td>.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t+1</td>
<td>H</td>
<td>0</td>
<td>.1</td>
<td>.9</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
<td>0</td>
<td>.1</td>
<td>.9</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.1</td>
</tr>
</tbody>
</table>

Conditional probability table (CPT)
Factored States & Transitions

- Each next-stage variable has a CPT
- This allows for a much more compact representation!
- “Two-stage dynamic Bayesian network” (2DBN)

```
L  H  K  M
L 1  .9 0  0
H 0  .1 .9 0
K 0  0 .1 .9
M 0  0 0  .1
```

classical probability table (CPT)
Factored States & Transitions

Do we always have so much independence? (what about other actions?)

Conditional probability table (CPT)

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>H</th>
<th>K</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>1</td>
<td>.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t+1</td>
<td>H</td>
<td>0</td>
<td>.1</td>
<td>.9</td>
</tr>
<tr>
<td></td>
<td>K</td>
<td>0</td>
<td>0</td>
<td>.1</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Factored States & Transitions

CPT encodes that IF
- \( \text{loc} = \text{lab} \)
- \( \text{CR} = 1 \)
→ high probability of CR becoming 0

“Deliver coffee”
Solving Factored MDPs

- CPT also representable as a decision tree

```
  RHC
   /  \
  0   1
 /  \  /
CR  CR
   /  \
  0   1
   /  \  /
 L   L
   /  \  /
loc
```

- \( P(CR' = 1) = 0.05 \)
- \( P(CR' = 0) = 0.95 \)
- \( P(CR' = 1) = 1 \)
- \( P(CR' = 0) = 0 \)
- \( P(CR' = 1) = 0.05 \)
- \( P(CR' = 0) = 0.95 \)
- \( P(CR' = 1) = 1 \)
- \( P(CR' = 0) = 0 \)
- \( P(CR' = 1) = 0.05 \)
- \( P(CR' = 0) = 0.95 \)
Solving Factored MDPs

- CPT also representable as a decision tree

![Decision Tree Diagram]

Similarly: rewards can be represented as decision trees (or ADDs)

→ So...?
Solving Factored MDPs

- CPT also representable as a decision tree

Similarly: rewards can be represented as decision trees (or ADDs)

→ Can also represent value functions, policies as decision trees [Boutilier et al 99]
Factored POMDPs

- Of course POMDP models can also be factored
- Similar ideas applied [Hansen & Feng 2000, Poupart 2005, Shani et al. 2008]
  - $\alpha$-vectors represented by ADDs
  - beliefs too.
- This does not solve all problems:
  - over time state factors get more and more correlated, so representation grows large.
Factored Multiagent Models

- Of course multiagent models can also be factored!
- Work can be categorized in a few directions:
  - Trying to execute the factored (PO)MDP policy
    [Roth et al. 2007, Messias et al. 2011]
  - Trying to execute independently as much as possible
    [Spaan & Melo 2008, Melo & Veloso 2011]
  - Exploiting graphical structure between agents
    (ND-POMDPs, Factored Dec-POMDPs)
  - Influence-based abstraction of policies of other agents
    (TOI-Dec-MDPs, TD-POMDPs, IBA for POSGs)
Graphical Structure between Agents

- Exploit (conditional) independence between agents
  - E.g., sensor networks [Nair et al '05 AAAI, Varakantham et al. '07 AAMAS]
Graphical Structure between Agents

- Exploit (conditional) independence between agents
  - E.g., sensor networks

These problems have
- State that cannot be influenced
- Factored reward function

\[ R(s,a) = \sum_e R_e(s,a_e) \]
Graphical Structure between Agents

- Exploit (conditional) independence between agents
  - E.g., sensor networks

These problems have
- State that cannot be influenced
- Factored reward function

\[ R(s, a) = \sum_e R_e(s, a_e) \]

This allows a reformulation as a (D)COP

\[ V(\pi) = \sum_e V_e(\pi_e) \]
Graphical Structure between Agents

- Exploit (conditional) independence between agents
  - E.g., sensor networks

These problems have
- State that cannot be influenced
- Factored reward function

\[ R(s, a) = \sum_e R_e(s, a_e) \]

This allows a reformulation as a (D)COP

\[ V(\pi) = \sum_e V_e(\pi_e) \]
Graphical Structure between Agents

- Factored Dec-POMDPs
  [Oliehoek et al. 2008 AAMAS]

\[
\begin{align*}
S^t & \quad S^{t+1} \\
\text{FL}_1 & \quad \text{FL}_1 \quad R_1 \\
\text{FL}_2 & \quad \text{FL}_2 \quad R_2 \\
\text{FL}_3 & \quad \text{FL}_3 \quad R_3 \\
\text{FL}_4 & \quad \text{FL}_4 \quad R_4 \\
\text{a}_1 & \quad \text{a}_1 & \quad \text{a}_2 & \quad \text{a}_3
\end{align*}
\]
Graphical Structure between Agents

- Factored Dec-POMDPs

Can't we use the previous methods (reduction to DCOP) directly...

- Why?
Graphical Structure between Agents

- Factored Dec-POMDPs

Can't we use the previous methods (reduction to DCOP) directly...
  - Why?
    → dependence propagates!

Can't we use the previous methods (reduction to DCOP) directly...
  - Why?
    → dependence propagates!
Graphical Structure between Agents

- Factored Dec-POMDPs

Can't we use the previous methods (reduction to DCOP) directly...
- Why?
  → dependence propagates!

Can't we use the previous methods (reduction to DCOP) directly...
- Why?
  → dependence propagates!
Graphical Structure between Agents

- Factored Dec-POMDPs

Can't we use the previous methods (reduction to DCOP) directly...

• Why?
  → dependence propagates!

Can't we use the previous methods (reduction to DCOP) directly...

● Why?
  → dependence propagates!

What influences $R_1^2$?
Graphical Structure between Agents

Factored Dec-POMDPs

Can't we use the previous methods (reduction to DCOP) directly...

- Why?
  - dependence propagates!

Can't we use the previous methods (reduction to DCOP) directly...

- Why?
  - dependence propagates!
Graphical Structure between Agents

- Factored Dec-POMDPs

Can't we use the previous methods (reduction to DCOP) directly...
  - Why?
    → dependence propagates!

![Graphical Structure Diagram]

- Can't we use the previous methods (reduction to DCOP) directly...
  - Why?
    → dependence propagates!
Graphical Structure between Agents

- Factored Dec-POMDPs

Can't we use the previous methods (reduction to DCOP) directly...

- Why?
  → dependence propagates!
Graphical Structure between Agents

- **Factored Dec-POMDPs**
  [Oliehoek et al. 2008 AAMAS]

- Solution Methods
  - reduction to a type of COP
  - but now: one for each stage!

- $\delta$ is a decision rule
  (part of policy for 1 stage $t$)

  $\rightarrow$ leads to factored form of heuristic search
  [Oliehoek 2010 PhD]
Influence-Based Abstraction

- Try to define agents' **local state**
- Analyze how policies of other agents affect it
  - find compact description for this influence

- Example: Mars Rovers [Becker et al. 2004 JAIR]
  - 2 rovers collect data at 4 sites
Influence-Based Abstraction

Transitions independent: Rovers drive independently
Rewards are dependent:

- 2 same soil samples of same site not so useful (sub additive)
- 2 pictures of (different sides) of same rock is useful (super additive)

Example: Mars Rovers [Becker et al. 2004 JAIR]

- 2 rovers collect data at 4 sites
Influence-Based Abstraction

- **TI Dec-MDP**
- extra reward (or penalty) at the end if 'joint event' happens
- joint event $E=<e_1, e_2>$
- From agent i's perspective: if it realizes $e_i$ → extra reward with probability $P(e_j)$
Influence-Based Abstraction

- **TI Dec-MDP**
  - extra reward (or penalty) at the end if 'joint event' happens
  - joint event $E = \langle e_1, e_2 \rangle$

But most problems are not transition independent!?

Much further research, e.g.:
- **Event-driven Dec-MDPs** [Becker et al.04 AAMAS]
- **Transition-decoupled POMDPs** [Witwicki 2011 PhD]
- **EDI-CR** [Mostafa & Lesser 2009 WIIAT]
- **IBA for Factored POSGs** [Oliehoek et al. 2012 AAAI]
References

References can be found on the tutorial website:
www.st.ewi.tudelft.nl/~mtjspaan/tutorialDMuU/

Further references can be found in

Available from http://people.csail.mit.edu/fao/