Planning Techniques for Robotics

Planning under Uncertainty: Minimax Formulation

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Uncertainty in Robotics

• So far our planners assumed no uncertainty
  - execution is perfect

<table>
<thead>
<tr>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₄</td>
<td>S₅</td>
<td>S₆</td>
</tr>
</tbody>
</table>

convert into a graph

search the graph for a least-cost path from $s_{\text{start}}$ to $s_{\text{goal}}$
Uncertainty in Robotics

• So far our planners assumed no uncertainty
  - execution is perfect

- Any deviations from the plan are dealt by re-planning
- Could be quite suboptimal and sometimes dangerous
  - planning a path along cliff does not take into account slippage
  - others examples???
Uncertainty in Robotics

In this chapter, we will discuss the concept of uncertainty as it applies to robotics. Uncertainty is a fundamental aspect of robotic systems, as they must operate in environments that are inherently unpredictable. This uncertainty can arise from various sources, including sensor noise, environmental factors, and the dynamic nature of the tasks they perform.

One important aspect of uncertainty in robotics is the ability to handle approximate preferences. In many real-world scenarios, it is not feasible or even possible to have complete certainty about a robot’s goals or the environment it is operating in. Therefore, robots need to be able to make decisions based on approximate preferences, which can help them to adapt to changing situations.

In this chapter, we will explore how approximate preferences can be integrated into robotic systems to improve their performance. We will also discuss how this approach can be applied to disambiguating human intentions in navigation, which is a challenging task that requires the robot to interpret and respond to the intentions of a human partner.

Bradford Neuman
Carnegie Mellon University Robotics Institute

Maxim Likhachev

Please turn on your audio.
Uncertainty in Robotics

• Modeling uncertainty in execution during planning

- at least one action in the graph has more than one outcome
- each outcome is associated with probability and cost

Markov Decision Processes (MDP)
- at least one action in the graph has more than one outcome
- each outcome is associated with probability and cost

example: $s_3, s_4, s_5 \in \text{succ}(s_2, a_{SE})$,

\[
\begin{align*}
P(s_3|a_{se},s_2) & = 0.9, \quad c(s_2,a_{se},s_5) = 1.4 \\
P(s_3|a_{se},s_2) & = 0.05, \quad c(s_2,a_{se},s_3) = 1.0 \\
P(s_4|a_{se},s_2) & = 0.05, \quad c(s_2,a_{se},s_4) = 1.0
\end{align*}
\]

Markov Decision Processes (MDP)
Moving along Cliff Example

- Example on the board
Moving-Target Search Example

• Uncertainty in the target moves

• What is a state-space and action space?
Planning in MDPs

• What plan to compute?
  - Plan that minimizes the worst-case scenario (minimax plan)
  - Plan that minimizes the expected cost

• Without uncertainty, plan is a single path:
  a sequence of states (a sequence of actions)
• In MDPs, plan is a policy $\pi$: 
  mapping from a state onto an action
Planning in MDPs

• What plan to compute?
  - Plan that minimizes the worst-case scenario (minimax plan)
  - Plan that minimizes the expected cost

• Without uncertainty, plan is a single path:
  a sequence of states (a sequence of actions)

• In MDPs, plan is a policy $\pi$:
  mapping from a state onto an action

Why?
Minimax Formulation

- Optimal policy $\pi^*$:
  minimizes the worst cost-to-goal
  
  $$\pi^* = \arg\min_\pi \max_{\text{outcomes of } \pi} \{\text{cost-to-goal}\}$$

- worst cost-to-goal for $\pi_1 = \text{(go through s}_4\text{)}$ is:
  
  $$1 + 1 + 3 + 1 = 6$$

- worst cost-to-goal for $\pi_2 = \text{(try to go through s}_1\text{)}$ is:
  
  $$1 + 2 + 2 + 2 + 2 + 2 + \ldots = \infty$$
Minimax Formulation

![Diagram](image)

- Optimal policy $\pi^*$:
  minimizes the worst cost-to-goal
  $\pi^* = \arg\min_\pi \max_{\text{outcomes of } \pi} \{\text{cost-to-goal}\}$

- Optimal minimax policy $\pi^* = (\text{go through } s_4) =$
  $\left[\{s_{\text{start}}, a_{ne}\}, \{s_2, a_{south}\}, \{s_4, a_{east}\}, \{s_3, a_{ne}\}, \{s_{\text{goal}}, \text{null}\}\right]$
Computing Minimax Plans

Minimax backward A*:

\[ g(s_{goal}) = 0; \text{ all other } g\text{-values are infinite}; \ OPEN = \{s_{goal}\}; \]

while (s_{start} not expanded)

remove s with the smallest \[ f(s) = g(s) + h(s) \] from \ OPEN; \n
insert s into \ CLOSED; \n
for every \ s'\ s.t s \in succ(s', a)\ for some a and s' not in \ CLOSED \n
if \ g(s') > \max_{u \in succ(s', a)} c(s',u) + g(u) \n
\[ g(s') = \max_{u \in succ(s', a)} c(s',u) + g(u); \]

insert \ s'\ into \ OPEN;
Computing Minimax Plans

- Minimax backward A*:
  
g(s_{goal}) = 0; all other g-values are infinite; OPEN = \{s_{goal}\};

while (s_{start} not expanded)
  remove s with the smallest \([f(s) = g(s) + h(s)]\) from OPEN;
  insert s into CLOSED;

for every \(s'\) s.t \(s \in \text{succ}(s', a)\) for some \(a\) and \(s'\) not in CLOSED
  if \(g(s') > \max_{u \in \text{succ}(s', a)} c(s', u) + g(u)\)
    \(g(s') = \max_{u \in \text{succ}(s', a)} c(s', u) + g(u)\);
  insert \(s'\) into OPEN;

reduces to usual backward A* if no uncertainty in outcomes
Computing Minimax Plans

Minimax backward A*:

- \( g(s_{goal}) = 0 \); all other \( g \)-values are infinite; \( OPEN = \{s_{goal}\} \);
- while (\( s_{start} \) not expanded)
  - remove \( s \) with the smallest \( f(s) = g(s) + h(s) \) from \( OPEN \);
  - insert \( s \) into \( CLOSED \);
  - for every \( s' \) s.t \( s \in succ(s', a) \) for some \( a \) and \( s' \) not in \( CLOSED \)
    - if \( g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u) \)
      - \( g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u) \);
      - insert \( s' \) into \( OPEN \);

\[
\begin{align*}
S_2: & \quad g = \infty \quad h=0 \\
S_1: & \quad g = \infty \quad h=2 \\
S_3: & \quad g = \infty \quad h=3 \\
S_4: & \quad g = \infty \quad h=2 \\
S_{start}: & \quad g = \infty \quad h=0 \\
S_{goal}: & \quad g = 0 \quad h=3
\end{align*}
\]
Computing Minimax Plans

- Minimax backward A*:
  \[ g(s_{\text{goal}}) = 0; \text{ all other } g\text{-values are infinite}; \text{ OPEN} = \{s_{\text{goal}}\}; \]
  \[ \text{while}(s_{\text{start}} \text{ not expanded}) \]
  \[ \text{remove } s \text{ with the smallest } [f(s) = g(s) + h(s)] \text{ from OPEN;} \]
  \[ \text{insert } s \text{ into CLOSED;}; \]
  \[ \text{for every } s' \text{ s.t } s \in \text{succ}(s', a) \text{ for some } a \text{ and } s' \text{ not in CLOSED} \]
  \[ \text{if } g(s') > \max_{u \in \text{succ}(s', a)} c(s', u) + g(u) \]
  \[ g(s') = \max_{u \in \text{succ}(s', a)} c(s', u) + g(u); \]
  \[ \text{insert } s' \text{ into OPEN;} \]

After \( s_{\text{goal}} \) expanded, what are \( g(s_3) \) and \( g(s_1) \)?
Computing Minimax Plans

- **Minimax backward A***:

  
  \[
  g(s_{\text{goal}}) = 0; \text{ all other } g\text{-values are infinite};\ OPEN = \{s_{\text{goal}}\};
  \]

  while \( s_{\text{start}} \text{ not expanded} \)

  remove \( s \) with the smallest \( [f(s) = g(s) + h(s)] \) from \( OPEN \);

  insert \( s \) into \( CLOSED \);

  for every \( s' \) s.t \( s \in \text{succ}(s', a) \) for some \( a \) and \( s' \) not in \( CLOSED \)

  if \( g(s') > \max_u \in \text{succ}(s', a) c(s', u) + g(u) \)

  \[
  g(s') = \max_u \in \text{succ}(s', a) c(s', u) + g(u);
  \]

  insert \( s' \) into \( OPEN \);
Computing Minimax Plans

- Minimax backward A*:
  
  \[ g(s_\text{goal}) = 0; \text{ all other } g\text{-values are infinite}; \quad OPEN = \{s_\text{goal}\}; \]
  
  while \( s_\text{start} \text{ not expanded} \)
  
  remove \( s \) with the smallest \([f(s) = g(s) + h(s)]\) from \( OPEN \);
  
  insert \( s \) into \( CLOSED \);
  
  for every \( s' \) s.t \( s \in succ(s',a) \) for some \( a \) and \( s' \) not in \( CLOSED \)
  
  if \( g(s') > \max_{u \in succ(s',a)} c(s',u) + g(u) \)
  
  \[ g(s') = \max_{u \in succ(s',a)} c(s',u) + g(u); \]
  
  insert \( s' \) into \( OPEN \);
Computing Minimax Plans

Minimax backward A*

- Minimax backward A*:
  - $g(s_{goal}) = 0$; all other $g$-values are infinite;
  - OPEN = \{s_{goal}\};
  - while (s_{start} not expanded)
    - remove $s$ with the smallest $[f(s) = g(s) + h(s)]$ from OPEN;
    - insert $s$ into CLOSED;
  - for every $s'$ s.t $s \in succ(s', a)$ for some $a$ and $s'$ not in CLOSED
    - if $g(s') > max_{u \in succ(s', a)} c(s', u) + g(u)$
      - $g(s') = max_{u \in succ(s', a)} c(s', u) + g(u)$;
      - insert $s'$ into OPEN;

$P(s_{goal}|s_1,a_1) = 0.9$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_1,a_1) = 0.1$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.2$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.3$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.4$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.5$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.6$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.7$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.8$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 0.9$
$c(s_1,a_1,s_{goal}) = 2$
$P(s_{goal}|s_2,a_1) = 1.0$
$c(s_1,a_1,s_{goal}) = 2$

CLOSED = \{s_{goal}, s_3, s_4\}
OPEN = \{s_2\}
next state to expand: $s_2$
Computing Minimax Plans

- **Minimax backward A**: 
  \[ g(s_{\text{goal}}) = 0; \text{ all other } g\text{-values are infinite}; \ OPEN = \{s_{\text{goal}}\}; \]

While \( s_{\text{start}} \) not expanded

  - remove \( s \) with the smallest \( f(s) = g(s) + h(s) \) from \( OPEN \);
  - insert \( s \) into \( CLOSED \);

  - for every \( s' \) s.t \( s \in \text{succ}(s', a) \) for some \( a \) and \( s' \) not in \( CLOSED \)
    - if \( g(s') > \max_{u \in \text{succ}(s', a)} c(s', u) + g(u) \)
      \[ g(s') = \max_{u \in \text{succ}(s', a)} c(s', u) + g(u); \]
    - insert \( s' \) into \( OPEN \);

\[ g = 0 \quad h = 3 \]
\[ OPEN = \{s_{\text{start}}, s_1\} \]
\[ CLOSED = \{s_{\text{goal}}, s_3, s_4, s_2\} \]
next state to expand: \( s_{\text{start}} \)
Computing Minimax Plans

- Minimax backward A*: 

  \[ g(s_{\text{goal}}) = 0; \text{ all other } g\text{-values are infinite}; \ OPEN = \{s_{\text{goal}}\}; \]

  while (\text{start not expanded})

  remove \( s \) with the smallest \([f(s) = g(s) + h(s)]\) from \( OPEN \);

  insert \( s \) into \( CLOSED \);

  for every \( s' \) s.t \( s \in \text{succ}(s', a) \) for some \( a \) and \( s' \) not in \( CLOSED \)

  if \( g(s') > \max_{u \in \text{succ}(s', a)} c(s', u) + g(u) \)

  \[ g(s') = \max_{u \in \text{succ}(s', a)} c(s', u) + g(u); \]

  insert \( s' \) into \( OPEN \);
Computing Minimax Plans

Minimax backward A*:

\[ g(s_{\text{goal}}) = 0; \text{all other } g\text{-values are infinite}; \]  
\[ \text{OPEN} = \{s_{\text{goal}}\}; \]

while (\(s_{\text{start}}\) not expanded)

remove \(s\) with the smallest \([f(s) = g(s) + h(s)]\) from OPEN;

insert \(s\) into CLOSED;

for every \(s'\) s.t \(s \in \text{succ}(s', a)\) for some \(a\) and \(s'\) not in CLOSED

if \(g(s') > \max_{u \in \text{succ}(s', a)} c(s', u) + g(u)\)

\[ g(s') = \max_{u \in \text{succ}(s', a)} c(s', u) + g(u); \]

insert \(s'\) into OPEN;

CLOSED = \(\{s_{\text{goal}}, s_3, s_4, s_2, s_{\text{start}}\}\)

OPEN = \(\{s_1\}\)

DONE

in this example, the computed policy is a path,  
but in general it is a tree
Computing Minimax Plans

- **Minimax backward A**: 
  
  
  
  $g(s_{goal}) = 0$; all other $g$-values are infinite; $OPEN = \{s_{goal}\}$; 
  while($s_{start}$ not expanded) 
  remove $s$ with the smallest $[f(s) = g(s)+h(s)]$ from $OPEN$; 
  insert $s$ into $CLOSED$; 
  for every $s'$ s.t $s \in succ(s', a)$ for some $a$ 
  if $g(s') > \max_{u \in succ(s', a)} c(s', u) + g(u)$ 
  $g(s') = \max_{u \in succ(s', a)} c(s', u) + g(u)$; 
  insert $s'$ into $OPEN$; 

  **DONE**

  **What are its branches?**

  **Why tree, and not graph?**

  *in this example, the computed policy is a path, but in general it is a tree*
Computing Minimax Plans

Minimax backward A*:

- $g(s_{\text{goal}}) = 0$; all other $g$-values are infinite; $OPEN = \{s_{\text{goal}}\}$;
- while ($s_{\text{start}}$ not expanded)
  - remove $s$ with the smallest $[f(s) = g(s)+h(s)]$ from $OPEN$;
  - insert $s$ into $CLOSED$;
- for every $s'$ s.t $s \in succ(s', a)$ for some $a$ and $s'$ not in $CLOSED$
  - if $g(s') > \max_{u \in succ(s', a)} c(s',u) + g(u)$
    - $g(s') = \max_{u \in succ(s', a)} c(s',u) + g(u)$
    - insert $s'$ into $OPEN$;

Minimax A* guarantees to find an optimal plan, and never expands a state more than once, provided heuristics are consistent (just like A*).
Computing Minimax Plans

• Pros/cons of minimax plans
  - robust to uncertainty
  - overly pessimistic
  - harder to compute than normal paths
    - especially if backwards minimax A* does not apply
    - even if backwards minimax A* does apply, still more expensive than computing a single path with A* (heuristics are not guiding well)

Why?
What You Should Know..

• What is and MDP (Markov Decision Process) and how it differs from normal Graphs

• What is Minimax solution to MDPs

• Pros and cons of Minimax solutions

• Operation of Minimax backward A*