Planning & Decision-making in 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

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

• Modeling uncertainty in execution during planning

Markov Decision Processes (MDP)

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

- Modeling uncertainty in execution during planning

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

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

- 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 $\pi^* = \text{(go through } s_4) = [\{s_{\text{start}}, a_{\text{ne}}\}, \{s_2, a_{\text{south}}\}, \{s_4, a_{\text{east}}\}, \{s_3, a_{\text{ne}}\}, \{s_{\text{goal}}, \text{null}\}]$
Minimax backward A*:

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

\[ \text{OPEN} = \{s_{\text{goal}}\};\]

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

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

\[ \text{insert } s \text{ into CLOSED};\]

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

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

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

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

- Minimax backward A*:
  
  \[ g(s_{goal}) = 0; \] all other \( g \)-values are infinite; \( OPEN = \{ s_{goal} \}; \]
  
  \[ \text{while(} s_{start} \text{ not expanded) } \]
  
  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 \( OPEN \);

\[ \text{CLOSED} = \{ \} \]
\[ OPEN = \{ s_{goal} \} \]
\[ \text{next state to expand: } s_{goal} \]

\[After \ s_{goal} \text{ expanded, what are } g(s_3) \text{ and } g(s_1) ?\]
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;
  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;

\( S_{start} \)
\[ g = \infty \]
\[ h = 0 \]

\( S_2 \)
\[ g = \infty \]
\[ h = 1 \]
\[ P(s_{goal}|s_1,a_1) = 0.9 \]
\[ c(s_1,a_1,s_{goal}) = 2 \]

\( S_1 \)
\[ g = \infty \]
\[ h = 2 \]
\[ P(s_2|s_1,a_1) = 0.1 \]
\[ c(s_1,a_1,s_2) = 2 \]

\( S_{goal} \)
\[ g = 0 \]
\[ h = 3 \]

\( S_3 \)
\[ g = 1 \]
\[ h = 3 \]

\( S_4 \)
\[ g = \infty \]
\[ h = 2 \]

CLOSED = \{s_{goal}\}
OPEN = \{s_3\}
next state to expand: s_3
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;
  next state to expand: \(s_4\)
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 $s \in \text{succ}(s', a)$ 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 $OPEN$;

$g = \infty$  
$h = 0$  

$g = \infty$  
$h = 1$  

$g = 5$  
$h = 2$  

$P(s_{\text{goal}} | s_1, a_1) = 0.9$

$c(s_1, a_1, s_{\text{goal}}) = 2$

$P(s_2 | s_1, a_1) = 0.1$

$c(s_1, a_1, s_2) = 2$

$g = 0$  
$h = 3$  

$g = \infty$  
$h = 2$  

$g = 4$  
$h = 2$  

$g = 1$  
$h = 3$  

$g = \infty$  
$h = 0$  

$CLOSED = \{s_{\text{goal}}, s_3, s_4\}$

$OPEN = \{s_2\}$

next state to expand: $s_2$
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 \( OPEN \);

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*:

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

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

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

\[ \text{insert } s \text{ into CLOSED}; \]

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

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

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

\[ \text{insert } s' \text{ into OPEN}; \]

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

OPEN = \{s_1\}

DONE
Computing Minimax Plans

- Minimax backward A*:

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

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

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

  \[ \text{insert } s \text{ into } CLOSED; \]

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

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

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

  \[ \text{insert } s' \text{ into } OPEN; \]
Computing Minimax Plans

Minimax backward A*:

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

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

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 \( a \)

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 \( Open; \)

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

\( Open = \{s_1\} \)

DONE

What are its branches? Why no cycles?

in this example, the computed policy is a path, but in general it is a Directed Acyclic Graph
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;

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

DONE

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

OPEN = \{s_1\}
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 Markov Decision Processes (MDP)

• Minimax formulation of planning under uncertainty

• The operation of Minimax backward A*

• Pros and cons of planning with Minimax formulation