15-887
Planning, Execution and Learning

Execution II: Real-time Heuristic Search
Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Planning during Execution

- Planning is a **repeated** process!
  - partially-known environments
  - dynamic environments
  - imperfect execution of plans
  - imprecise localization

- Need to be able to re-plan fast!

- Several methodologies to achieve this:
  - anytime heuristic search: return the best plan possible within T msecs
  - incremental heuristic search: speed up search by reusing previous efforts
  - real-time heuristic search: plan few steps towards the goal and re-plan later

*this class*
Real-time (Agent-centered) Heuristic Search

Enforce a strict limit on the amount of computations (no requirement on planning all the way to the goal)
Real-time (Agent-centered) Heuristic Search

1. Compute a partial path by expanding at most N states around the robot

2. Move once, incorporate sensor information, and goto step 1

Example in a fully-known terrain:

- expanded
Real-time (Agent-centered) Heuristic Search

1. Compute a partial path by expanding at most N states around the robot
2. Move once, incorporate sensor information, and goto step 1

Example in an unknown terrain (planning with Freespace Assumption):

- expanded
Planning with Freespace Assumption [Nourbakhsh & Genesereth, ‘96]

• Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path in a partially-known terrain to the goal state.

• Replan the path whenever a new sensor information received.

_costs between unknown states is the same as the costs in between states known to be free_
Planning with Freespace Assumption [Nourbakhsh & Genesereth, ‘96]

• Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path in a partially-known terrain to the goal state

• Replan the path whenever a new sensor information received
Real-time (Agent-centered) Heuristic Search

1. Compute a partial path by expanding at most N states around the robot
2. Move once, incorporate sensor information, and goto step 1

Research issues:
- how to compute partial path
- how to guarantee complete behavior (guarantee to reach the goal)
- provide bounds on the number of steps before reaching the goal
Real-time (Agent-centered) Heuristic Search

1. Compute a partial path by expanding at most N states around the robot
2. Move once, incorporate sensor information, and goto step 1

Research issues:
- how to compute partial path Any ideas?
- how to guarantee complete behavior (guarantee to reach the goal)
- provide bounds on the number of steps before reaching the goal
Learning Real-Time A* (LRTA*) [Korf, ‘90]

- Repeatedly move the robot to the most promising adjacent state, using heuristics

  1. *always move as follows:* $s_{\text{start}} = \arg\min_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s)$

  $h(x,y) = \max(\text{abs}(x-x_{\text{goal}}), \text{abs}(y-y_{\text{goal}})) + 0.4 \times \min(\text{abs}(x-x_{\text{goal}}), \text{abs}(y-y_{\text{goal}}))$

- Any problems?
Learning Real-Time A* (LRTA*) [Korf, ‘90]

- Repeatedly move the robot to the most promising adjacent state, using heuristics

1. _always move as follows:_ \( s_{\text{start}} = \arg\min_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s) \)

\[
\begin{align*}
    h(x,y) &= \max(\text{abs}(x-x_{\text{goal}}), \text{abs}(y-y_{\text{goal}})) + 0.4 \times \min(\text{abs}(x-x_{\text{goal}}), \text{abs}(y-y_{\text{goal}}))
\end{align*}
\]

Local minima problem (myopic or incomplete behavior)

Any solutions?
Learning Real-Time A* (LRTA*) [Korf, ‘90]

- Repeatedly move the robot to the most promising adjacent state, using **and updating** heuristics

1. \[ h(s_{start}) = \min_{s \in \text{succ}(s_{start})} c(s_{start}, s) + h(s) \]
2. always move as follows: \[ s_{start} = \arg\min_{s \in \text{succ}(s_{start})} c(s_{start}, s) + h(s) \]

\[ \begin{array}{cccccc}
6.2 & 5.2 & 4.2 & 3.8 & 3.4 & 3 \\
5.8 & 4.8 & 3.8 & 2.8 & 2.4 & 2 \\
5.4 & 4.4 & 3.4 & 2.4 & 1.4 & 1 \\
5 & 4 & 3 & 2 & 1 & 0 \\
\end{array} \]

\[ \begin{array}{cccccc}
6.2 & 5.2 & 4.2 & 3.8 & 3.4 & 3 \\
5.8 & 4.8 & 3.8 & 2.8 & 2.4 & 2 \\
5.4 & 4.4 & 3.4 & 2.4 & 1.4 & 1 \\
5 & 4 & 3 & 2 & 1 & 0 \\
\end{array} \]

\[ \begin{array}{cccccc}
6.2 & 5.2 & 4.2 & 3.8 & 3.4 & 3 \\
5.8 & 4.8 & 3.8 & 2.8 & 2.4 & 2 \\
5.4 & 4.4 & 1.4 & 1 & 1 & 1 \\
5 & 4 & 1 & 0 & 0 & 0 \\
\end{array} \]

*makes h-values more informed*
Learning Real-Time A* (LRTA*) [Korf, ‘90]

- Repeatedly move the robot to the most promising adjacent state, using **and updating** heuristics

1. \( \text{update } h(s_{start}) = \min_{s \in \text{succ}(s_{start})} c(s_{start}, s) + h(s) \)
2. \( \text{always move as follows: } s_{start} = \arg\min_{s \in \text{succ}(s_{start})} c(s_{start}, s) + h(s) \)

\[
\begin{array}{ccccccc}
6.2 & 5.2 & 4.2 & 3.8 & 3.4 & 3 & \\
5.8 & 4.8 & 3.8 & 2.8 & 2.4 & 2 & \\
5.4 & 4.4 & 5 & 1.4 & 1 & 0 & \\
5 & 5.4 & 5 & 1 & 0 & & \\
\end{array}
\]

\[
\begin{array}{ccccccc}
6.2 & 5.2 & 4.2 & 3.8 & 3.4 & 3 & \\
5.8 & 4.8 & 3.8 & 2.8 & 2.4 & 2 & \\
5.4 & 4.4 & 5 & 1.4 & 1 & 0 & \\
5 & 5.4 & 5 & 1 & 0 & & \\
\end{array}
\]

\[
\begin{array}{ccccccc}
6.2 & 5.2 & 4.2 & 3.8 & 3.4 & 3 & \\
5.8 & 4.8 & 3.8 & 2.8 & 2.4 & 2 & \\
5.4 & 4.4 & 5 & 1.4 & 1 & 0 & \\
5 & 5.4 & 5 & 1 & 0 & & \\
\end{array}
\]
Learning Real-Time A* (LRTA*) [Korf, '90]

- Repeatedly move the robot to the most promising adjacent state, using **and updating** heuristics

1. \( \text{update } h(s_{\text{start}}) = \min_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s) \)
2. \( \text{always move as follows: } s_{\text{start}} = \arg\min_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s) \)

\( h\)-values remain admissible and consistent

**proof?**
Learning Real-Time A* (LRTA*) [Korf, '90]

- Repeatedly move the robot to the most promising adjacent state, using **and updating** heuristics

1. \( update \ h(s_{start}) = \min_{s \in succ(s_{start})} c(s_{start}, s) + h(s) \)
2. \( always \ move \ as \ follows: \ s_{start} = \arg\min_{s \in succ(s_{start})} c(s_{start}, s) + h(s) \)

<table>
<thead>
<tr>
<th></th>
<th>6.2</th>
<th>5.2</th>
<th>4.2</th>
<th>3.8</th>
<th>3.4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.8</td>
<td>4.8</td>
<td>3.8</td>
<td>2.8</td>
<td>2.4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5.4</td>
<td>4.4</td>
<td>1.4</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>6.2</th>
<th>5.2</th>
<th>4.2</th>
<th>3.8</th>
<th>3.4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.8</td>
<td>4.8</td>
<td>3.8</td>
<td>2.8</td>
<td>2.4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5.4</td>
<td>4.4</td>
<td>1.4</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>6.2</th>
<th>5.2</th>
<th>4.2</th>
<th>3.8</th>
<th>3.4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.8</td>
<td>4.8</td>
<td>3.8</td>
<td>2.8</td>
<td>2.4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5.4</td>
<td>4.4</td>
<td>1.4</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

robot is guaranteed to reach goal in finite number of steps if:

- all costs are bounded from below with \( \Delta > 0 \)
- graph is of finite size and there exists a finite-cost path to the goal
- all actions are reversible
Learning Real-Time A* (LRTA*) [Korf, ’90]

- Repeatedly move the robot to the most promising adjacent state, using and updating heuristics

1. update \( h(s_{\text{start}}) = \min_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s) \)

2. always move as follows: \( s_{\text{start}} = \text{argmin}_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s) \)

---

<table>
<thead>
<tr>
<th>6.2</th>
<th>5.2</th>
<th>4.2</th>
<th>3.8</th>
<th>3.4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8</td>
<td>4.8</td>
<td>3.8</td>
<td>2.8</td>
<td>2.4</td>
<td>2</td>
</tr>
<tr>
<td>5.4</td>
<td>4.4</td>
<td>1.4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5.4</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

robot is guaranteed to reach goal in finite number of steps if:

- all costs are bounded from below with \( \Delta > 0 \)
- graph is of finite size and there exists a finite-cost path to the goal
- all actions are reversible

Why conditions?
Learning Real-Time A* (LRTA*)

- **LRTA* with $N \geq 1$ expands** [Koenig, ‘04]

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$

- expanded
Learning Real-Time A* (LRTA*)

- **LRTA** with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \text{argmin}_{s' \in OPEN} g(s') + h(s')$

**state $s$:**

- the state that minimizes cost to it plus heuristic estimate of the remaining distance
- the state that looks most promising in terms of the whole path from current robot state to goal

- expanded
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

4-connected grid (robot moves in 4 directions)

example borrowed from ICAPS’06 planning summer school lecture (Koenig & Likhachev)

- expanded
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$
Learning Real-Time A* (LRTA*)

- **LRTA* with** $N \geq 1$ expands

1. **expand** $N$ states
2. **update** $h$-values of expanded states by Dynamic Programming (DP)
3. **move on the path to state** $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$

**How path is found?**

---

expand $N=7$ states

unexpanded state with smallest $g + h = 5 + 3$
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$

update $h$-values of expanded states via DP:
compute $h(s) = \min_{s' \in \text{succ}(s)} (c(s,s') + h(s'))$
until convergence

- expanded
Learning Real-Time A* (LRTA*)

• LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$

update $h$-values of expanded states via DP:
compute $h(s) = \min_{s' \in \text{succ}(s)} (c(s,s') + h(s'))$
until convergence

- expanded
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$

update $h$-values of expanded states via DP: compute $h(s) = \min_{s' \in \text{succ}(s)} (c(s,s') + h(s'))$ until convergence

- expanded
Learning Real-Time A* (LRTA*)

- **LRTA** with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg \min_{s' \in OPEN} g(s') + h(s')$

**Diagram:**

```
   8  7  6  5  4
   7  6  5  4  3
   6  5  4  3  2
∞  6  -  2  1
∞  ∞  ∞  0
```

update $h$-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s') + h(s'))$
until convergence

- expanded
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. *expand N states*
2. *update h-values of expanded states by Dynamic Programming (DP)*
3. *move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')*

update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s, s') + h(s'))$
until convergence

Does it matter in what order?
Learning Real-Time A* (LRTA*)

- **LRTA** with $N \geq 1$ expands
  
  1. expand $N$ states
  2. update $h$-values of expanded states by Dynamic Programming (DP)
  3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

update $h$-values of expanded states via DP:
compute $h(s) = \min_{s' \in \text{succ}(s)} (c(s,s') + h(s'))$
until convergence

- expanded
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

  1. expand $N$ states
  2. update $h$-values of expanded states by Dynamic Programming (DP)
  3. move on the path to state $s = \text{argmin}_{s' \in \text{OPEN}} g(s') + h(s')$

update $h$-values of expanded states via DP:
compute $h(s) = \min_{s' \in \text{succ}(s)} (c(s,s') + h(s'))$
until convergence
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

update $h$-values of expanded states via DP:
compute $h(s) = \min_{s' \in \text{succ}(s)} (c(s, s') + h(s'))$
until convergence

- expanded
Learning Real-Time A* (LRTA*)

- LRTA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states by Dynamic Programming (DP)
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

Drawbacks compared to A*?
Real-time Adaptive A* (RTAA*) [Koenig & Likhachev, ‘06]

- RTAA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states $u$ by $h(u) = f(s) - g(u)$,
   where $s = \arg\min_{s' \in OPEN} g(s') + h(s')$
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

expand $N=7$ states

unexpanded state $s$ with smallest $g + h (= 5 + 3)$

- expanded
Real-time Adaptive A* (RTAA*)

- RTAA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states $u$ by $h(u) = f(s) - g(u)$, where $s = \text{argmin }_{s' \in \text{OPEN}} g(s') + h(s')$
3. move on the path to state $s = \text{argmin }_{s' \in \text{OPEN}} g(s') + h(s')$

(update all expanded states $u$:
$h(u) = f(s) - g(u)$)

unexpanded state $s$ with smallest $f(s) = 8$
Real-time Adaptive A* (RTAA*)

- RTAA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states $u$ by $h(u) = f(s) - g(u)$, where $s = \arg\min_{s' \in OPEN} g(s') + h(s')$
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

update all expanded states $u$: $h(u) = f(s) - g(u)$

unexpanded state $s$ with smallest $f(s) = 8$
Real-time Adaptive A* (RTAA*)

- RTAA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states $u$ by $h(u) = f(s) - g(u)$, where $s = \arg\min_{s' \in OPEN} g(s') + h(s')$
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

update all expanded states $u$: $h(u) = f(s) - g(u)$

unexpanded state $s$ with smallest $f(s) = 8$

- expanded
Real-time Adaptive A* (RTAA*)

- RTAA* with $N \geq 1$ expands

1. expand $N$ states
2. update $h$-values of expanded states $u$ by $h(u) = f(s) - g(u)$,
   
   where $s = \text{argmin}_{s' \in \text{OPEN}} g(s') + h(s')$
3. move on the path to state $s = \text{argmin}_{s' \in \text{OPEN}} g(s') + h(s')$ 

proof of admissibility:

- $g(u) + h^*(u) \geq h^*(s_{\text{start}})$
- $h^*(u) \geq h^*(s_{\text{start}}) - g(u)$
- $f(s) \leq h^*(s_{\text{start}})$
- $h^*(u) \geq h_{\text{updated}}(u)$

\[ h^*() = \text{true cost-to-goal} \]

- expanded
LRTA* vs. RTAA*

LRTA*

<table>
<thead>
<tr>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>8</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

RTAA*

<table>
<thead>
<tr>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

- Update of $h$-values in RTAA* is much faster but not as informed
- Both guarantee admissibility and consistency of heuristics
- For both, heuristics are monotonically increasing
- Both guarantee to reach the goal in a finite number of steps (given the conditions listed previously)
Summary

• Real-time Heuristic Search puts a hard constraint on planning time (usually, a smaller planning time than what is required to plan a path all the way to the goal)

• Computing a partial path to the goal may result in highly sub-optimal behavior

• It is important to think how to avoid infinite oscillations
  – Updating heuristics is a popular way for doing it
  – Mostly applicable to low-dimensional planning
  – How to extend it to high-dimensional planning is a research question