16-782
Planning & Decision-making in Robotics

Interleaving Planning & Execution: 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 lecture
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]

• **Freespace Assumption**: all unknown cells are assumed to be traversable

• **Planning with the Freespace Assumption**: always move the robot on a shortest path to the goal assuming all unknown cells are traversable

• **Replan the path whenever a new sensor information received**

\[ \text{costs between unknown states is the same as the costs in between states known to be free} \]
Planning with Freespace Assumption [Nourbakhsh & Genesereth, ‘96]

- **Freespace Assumption**: all unknown cells are assumed to be traversable

- **Planning with the Freespace Assumption**: always move the robot on a shortest path to the goal assuming all unknown cells are traversable

- 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
- how to guarantee complete behavior (guarantee to reach the goal)
- provide bounds on the number of steps before reaching the goal

Any ideas?
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: $s_{start} = \arg\min_{s \in \text{succ}(s_{start})} c(s_{start}, s) + h(s)$

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

Local minima problem (myopic or incomplete behavior

\[\text{Any solutions?}\]
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: $s_{\text{start}} = \arg\min_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s)$
Learning Real-Time A* (LRTA*) [Korf, ‘90]

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

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

\(h\)-values remain admissible and consistent

\textit{proof?}
Learning Real-Time A* (LRTA*) [Korf, ‘90]

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

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

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

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

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

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

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: \( s_{\text{start}} = \arg\min_{s \in \text{succ}(s_{\text{start}})} c(s_{\text{start}}, s) + h(s) \)

\[ \begin{array}{cccccc}
6.2 & 5.2 & 4.2 & 3.8 & 3.4 & 3 \\
5.8 & 4.8 & 3.8 & 2.8 & 2.4 & 2 \\
5.4 & 4.4 & \textbf{5.2} & 1.4 & 1 & 0 \\
5 & 5.4 & 5 & \textbf{1.0} & 0 & 0 \\
\end{array} \]

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

necessary for the guarantee to reach the goal
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 $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

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)
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 $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

```
expand N=7 states
```

- 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 $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

---

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 $s = \arg \min_{s' \in \text{OPEN}} g(s') + h(s')$

How path is found?

- expand $N=7$ states
- unexpanded state with smallest $g + h (= 5 + 3)$

- 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 $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$
4. update $h$-values of expanded states via DP:
   \[
   h(s) = \min_{s' \in \text{succ}(s)} (c(s,s') + h(s'))
   \]
   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

Does it matter in what order?

- 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 $h(s) = \min_{s' \in \text{succ}(s)} (c(s,s') + h(s'))$
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
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 $s = \operatorname{argmin}_{s' \in \text{OPEN}} g(s') + h(s')$

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

- 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 $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$

make a move along the found path and repeat steps 1-3

Drawbacks compared to A*?

- expanded
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 $s = \text{argmin}_{s' \in OPEN} g(s') + h(s')$
3. move on the path to state $s = \text{argmin}_{s' \in OPEN} g(s') + h(s')$

```plaintext
expand N=7 states
```

- expanded

unexpanded state $s$ with smallest $g + h$ (= 5 + 3)
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
   \[
s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')
   \]
3. move on the path to state
   \[
s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')
   \]

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 $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$
3. move on the path to state $s = \arg\min_{s' \in \text{OPEN}} g(s') + h(s')$

![Matrix diagram]

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 $s = \arg\min_{s' \in OPEN} g(s') + h(s')$
3. move on the path to state $s = \arg\min_{s' \in OPEN} g(s') + h(s')$

proof of admissibility:

$$g(u) + h^*(u) \geq h^*(s_{start})$$

$$h^*(u) \geq h^*(s_{start}) - g(u)$$

$$h^*(u) \geq f(s) - g(u)$$

$$h^*(u) \geq h_{updated}(u)$$
LRTA* vs. RTAA*

- 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)
What You Should Know…

- What Freespace Assumption means

- Why we need to update heuristics in the context of Real-time Heuristic Search

- The operation of LRTA*

- Pros and cons of LRTA*

- What domains LRTA* is useful in and what domains it is not really applicable

- What RTAA* is