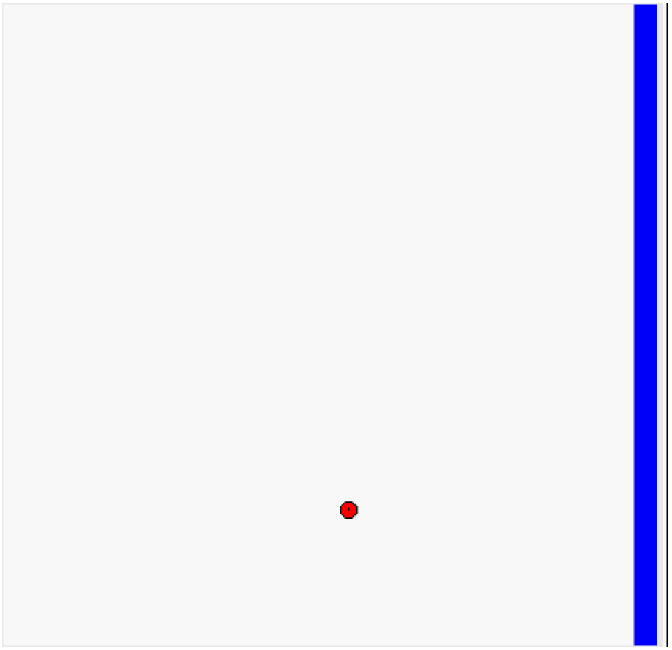
Real-time Planning and Re-planning II: Planning with Freespace Assumption, Agent-centered Search

Maxim Likhachev

Carnegie Mellon University

Planning with Freespace Assumption

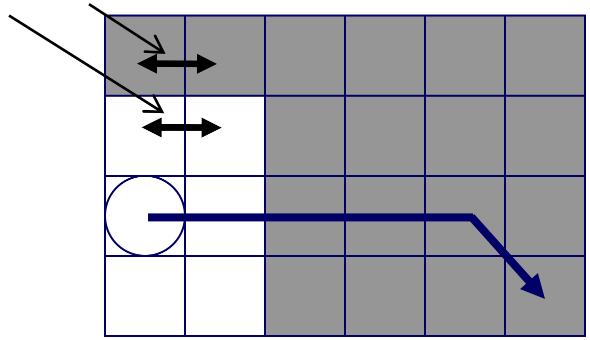
- Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path in a partially-known terrain to the goal cell
- Replan the path whenever a new sensor information received



Planning with Freespace Assumption

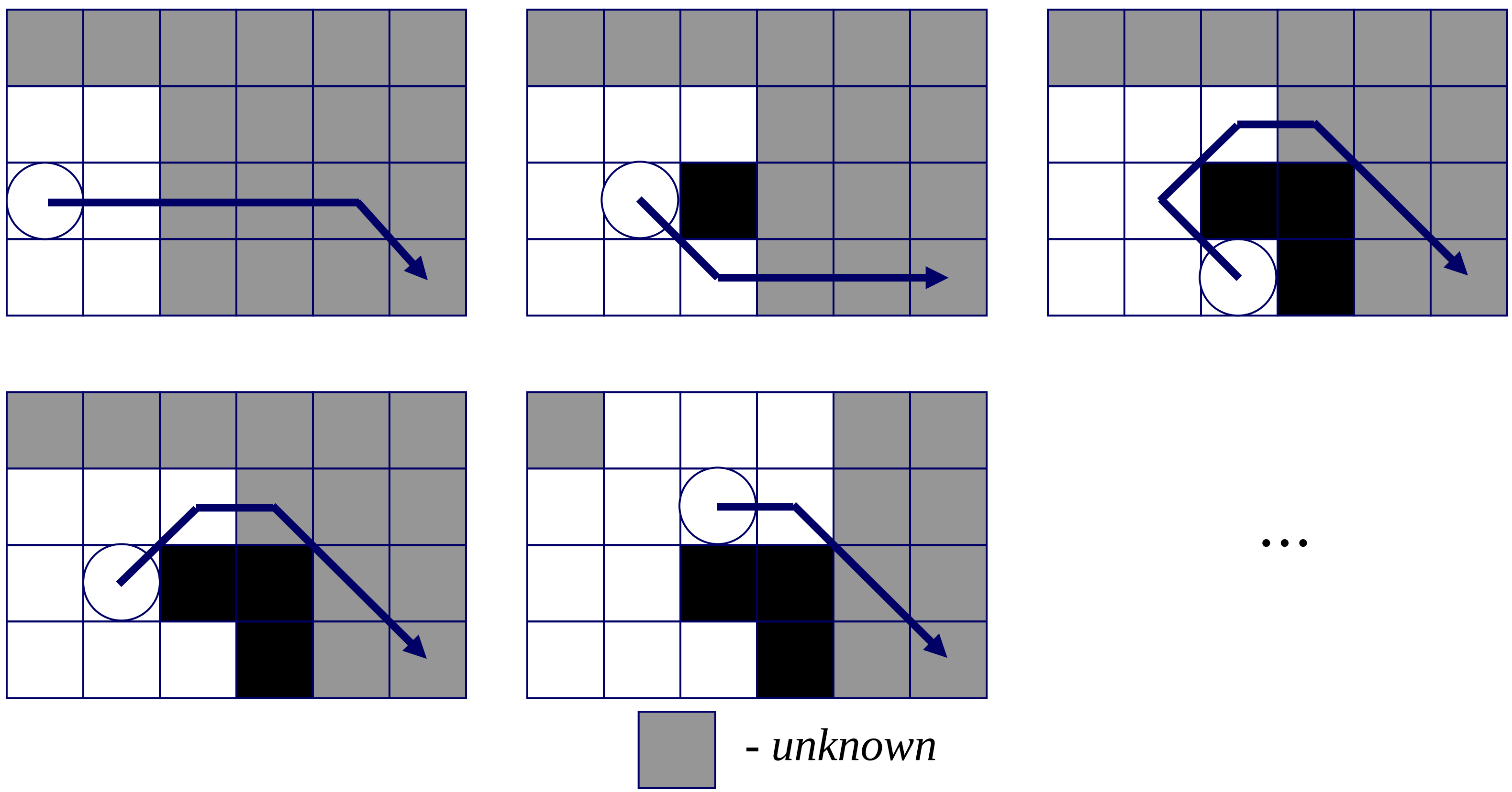
- Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path in a partially-known terrain to the goal cell
- Replan the path whenever a new sensor information received

costs between unknown cells is the same as the costs in between cells known to be free



Planning with Freespace Assumption

- Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path in a partially-known terrain to the goal cell
- Replan the path whenever a new sensor information received



Planning with Freespace Assumption

- Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path in a partially-known terrain to the goal cell
- Replan the path whenever a new sensor information received

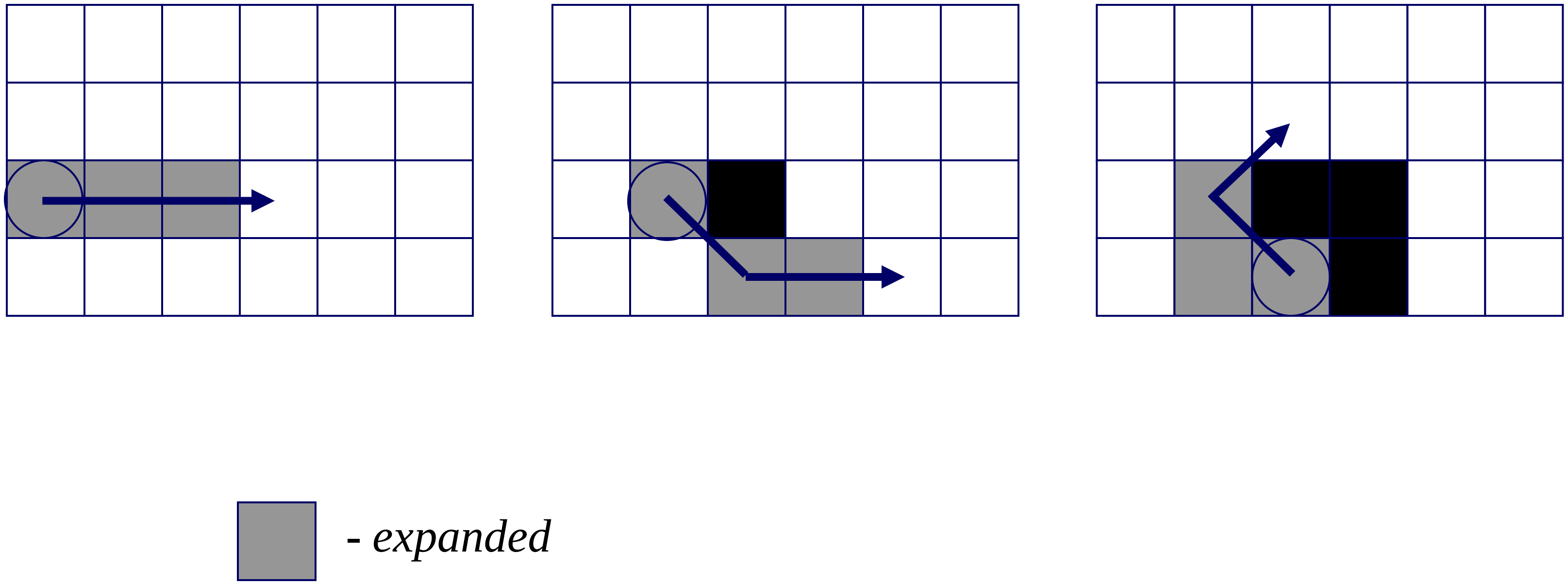
- A lot of replanning!
- Incremental planning helps (D*/D* Lite)
- Anytime planning helps (ARA*, Anytime D*)

- Agent-centered planning (this class):
  - a strict limit on the amount of computations (no planning all the way to the goal)

Agent-centered 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 (agent-centered search with the freespace assumption):

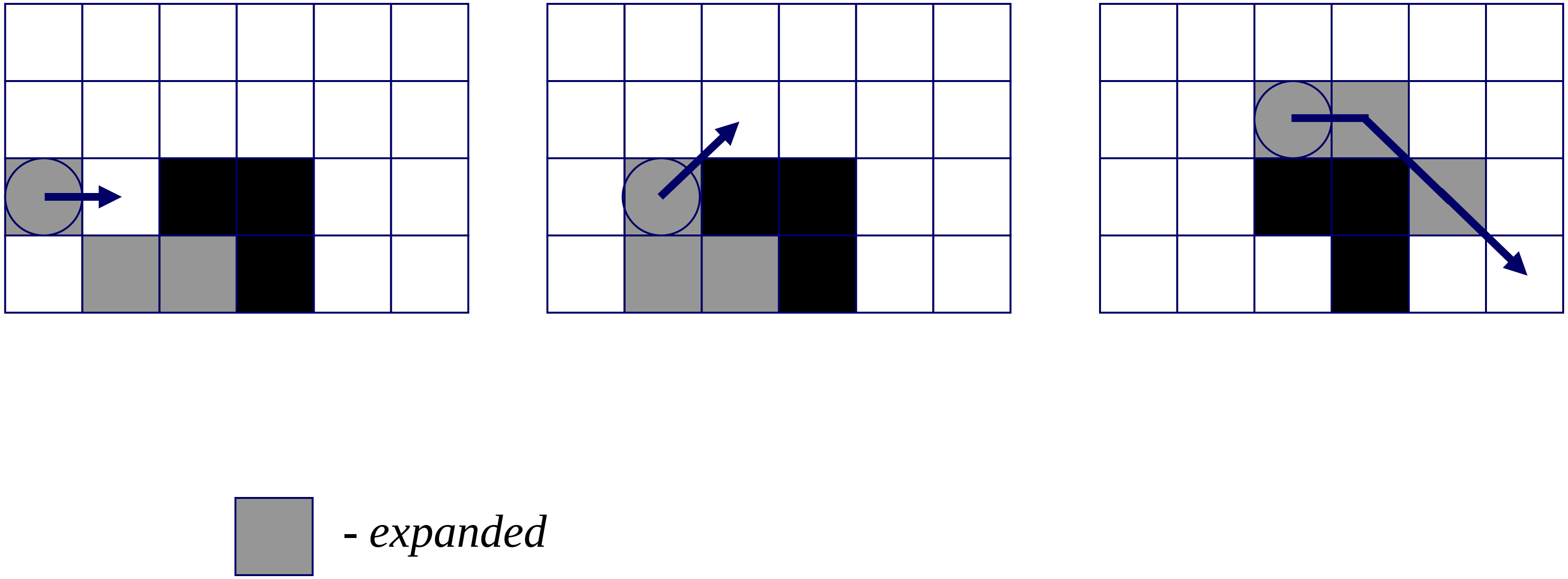


- *expanded*

Agent-centered 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*

Agent-centered 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

Agent-centered 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)

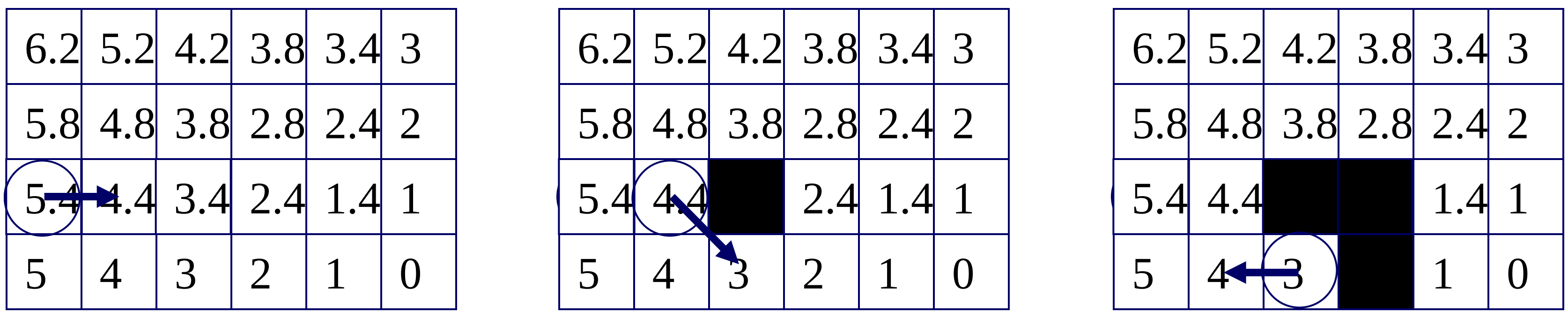
- provide bounds on the number of steps before reaching the goal

Learning Real-Time A* (LRTA*)

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

1. *always move as follows:* $s_{start} = \mathrm{argmin}_{s \in succ(sstart)} c(s_{start}, s) + h(s)$

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

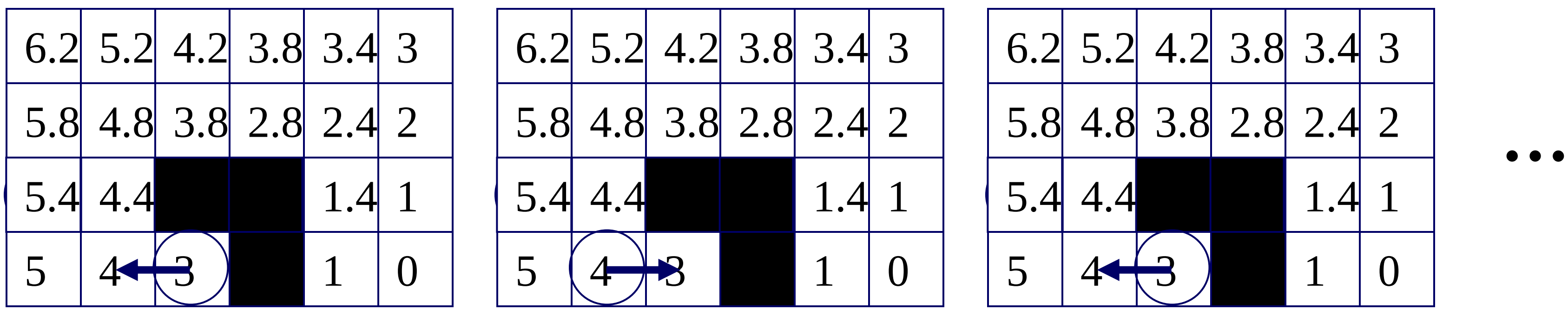


Any problems?

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

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

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



Local minima problem (myopic or incomplete behavior)

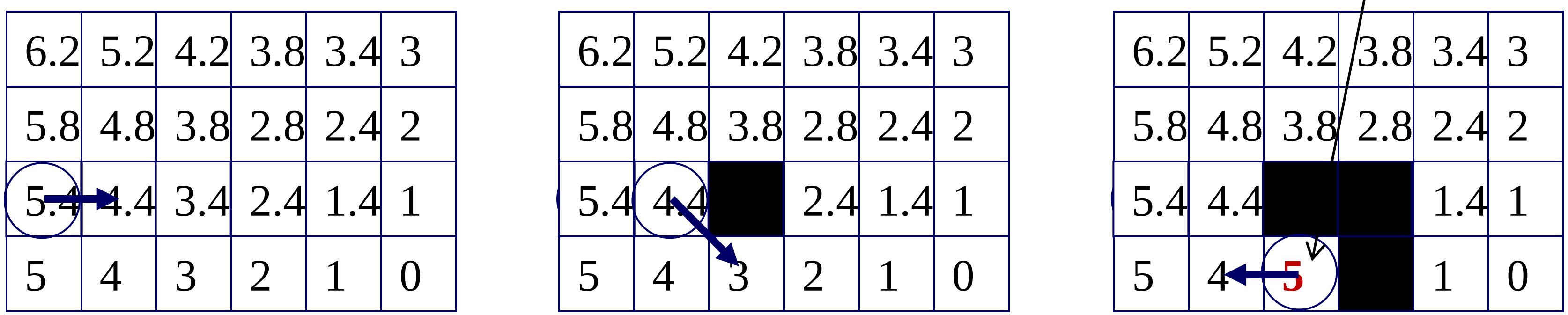
Any solutions?

Learning Real-Time A* (LRTA*)

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

makes h-values more informed

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

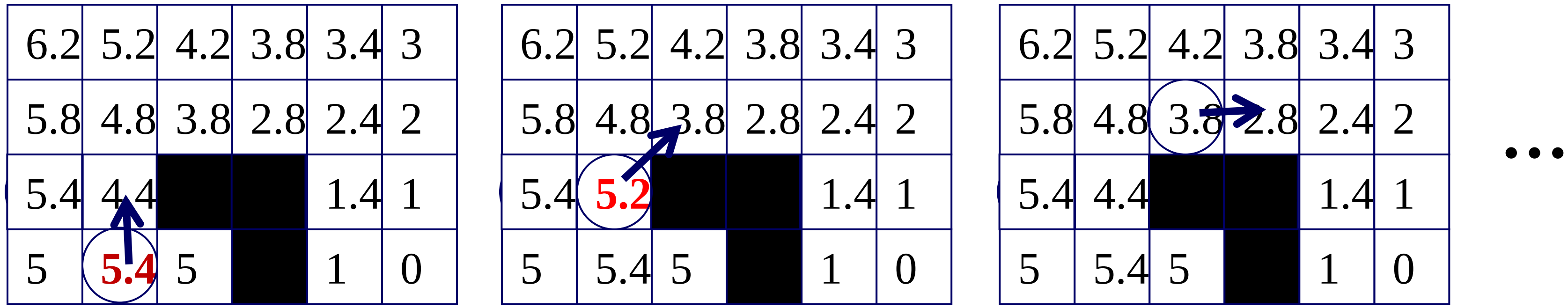


Learning Real-Time A* (LRTA*)

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

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

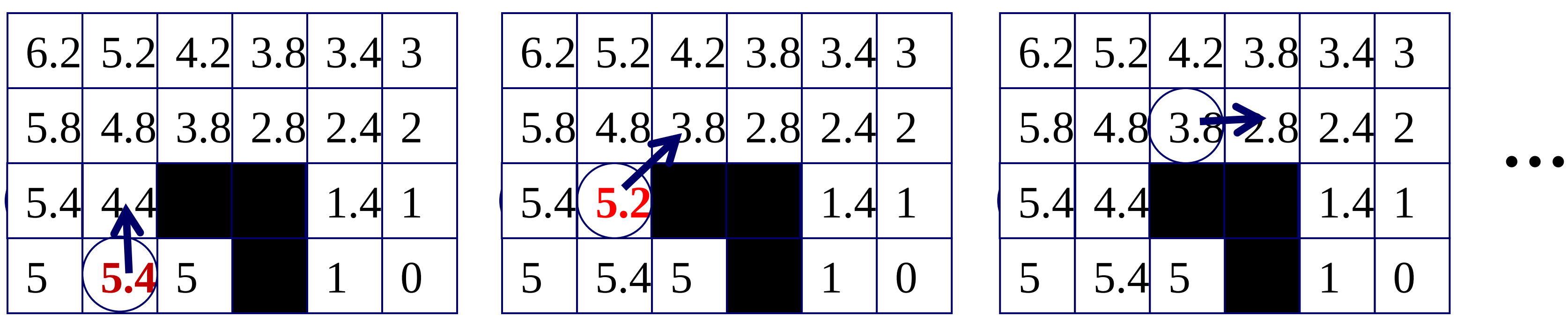
2. always move as follows: $s_{start} = \arg\min_{s \in succ(sstart)} c(s_{start}, s) + h(s)$



Learning Real-Time A* (LRTA*)

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

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



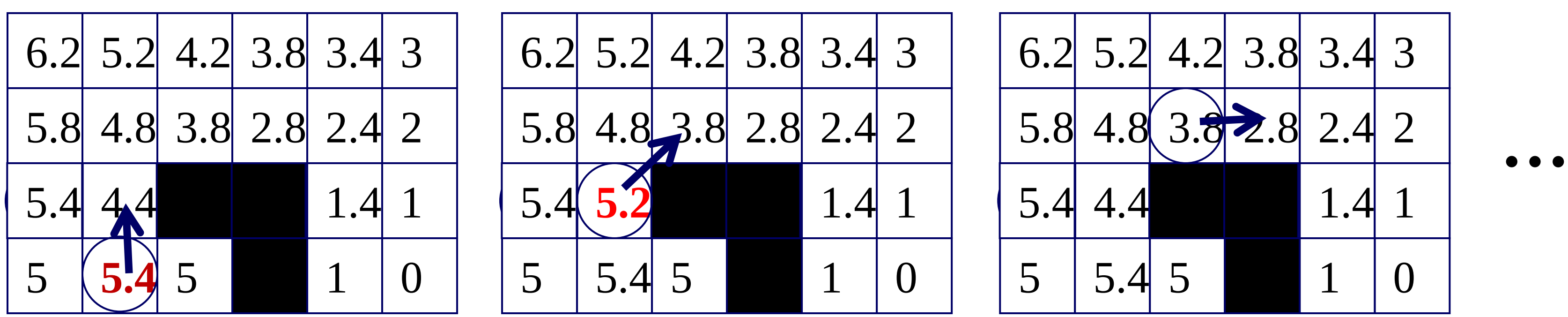
h-values remain admissible and consistent

proof?

Learning Real-Time A* (LRTA*)

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

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



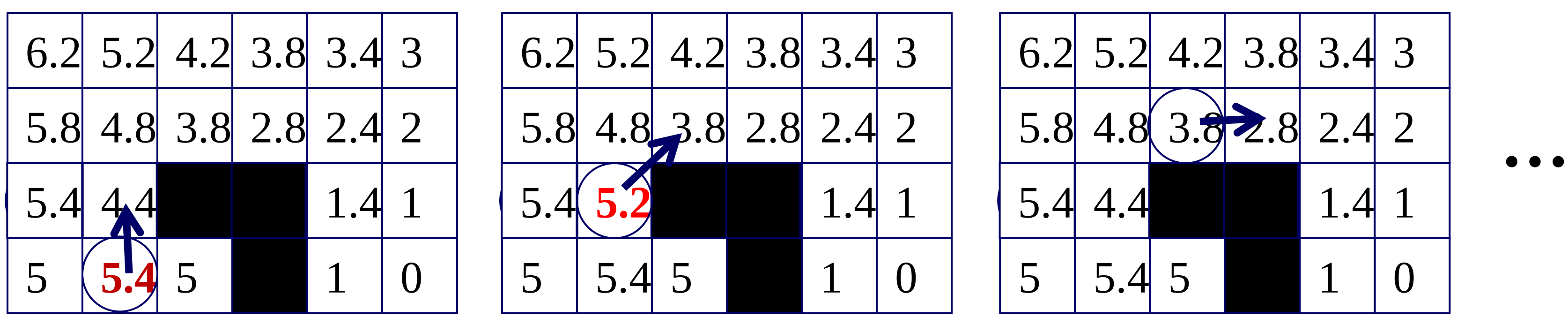
robot is guaranteed to reach goal in finite number of steps if:

- 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*)

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

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



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?

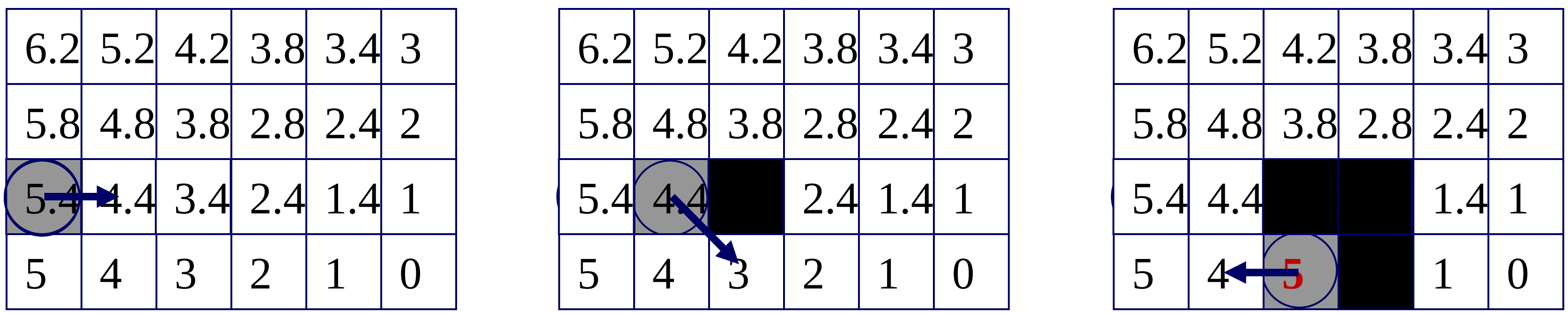
Learning Real-Time A* (LRTA*)

- related to limited-horizon A*:
  - expand $N = 1$ state, make a move towards a state s in OPEN with smallest $g(s)+h(s)$

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

   $= argmin_{\ s \in succ(sstart)} g(s) + h(s)$

Why?



- *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* $s = \text{argmin}_{s' \in OPEN}\, g(s') + h(s')$

necessary for the guarantee to reach the goal

How path is found?

- *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*:

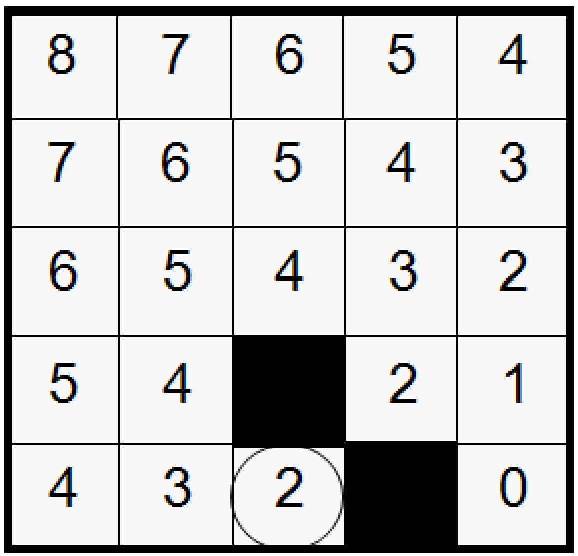
- the state that minimizes cost to it plus heuristic estimate of the remaining distance

- the state that looks most promising in terms of the whole path from current robot state to goal

- *expanded*

- 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)

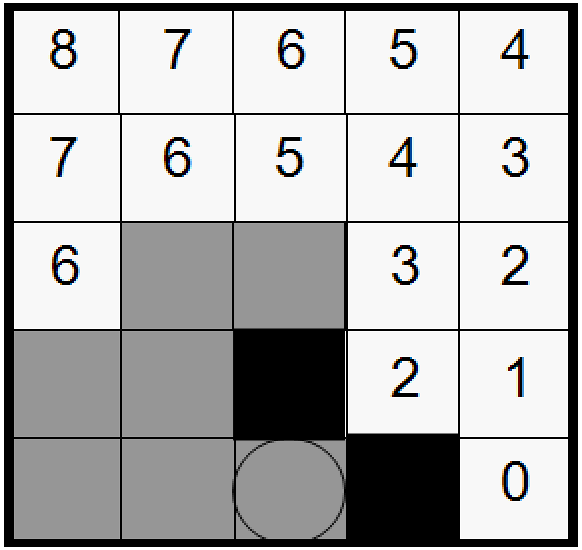
example borrowed from ICAPS'06 planning summer school lecture (Koenig & Likhachev)

- *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* $s = \text{argmin}_{s' \in OPEN}\, g(s') + h(s')$



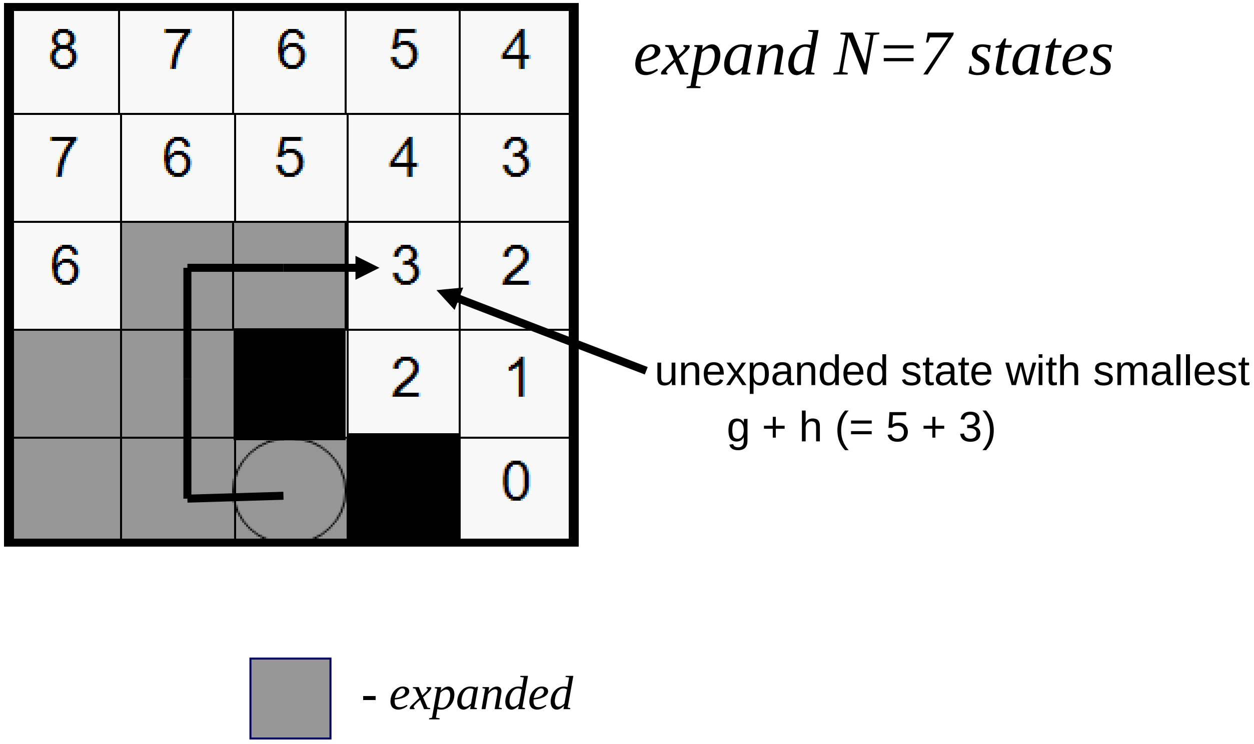
expand N=7 states

- *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* $s = \text{argmin}_{s' \in OPEN}\, g(s') + h(s')$

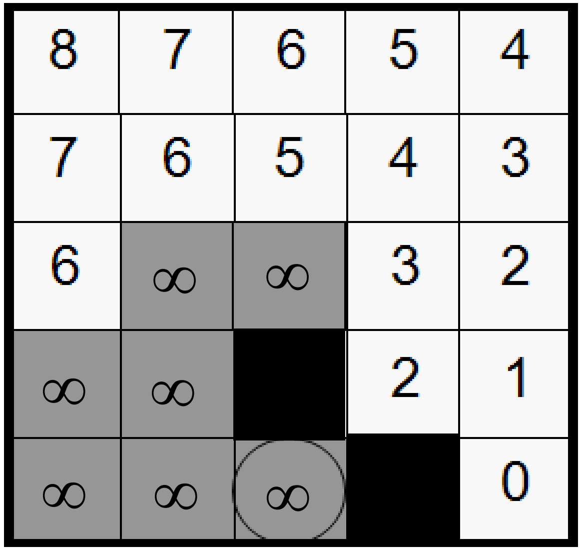


expand N=7 states

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 $s = \text{argmin}_{s' \in OPEN}\, g(s') + h(s')$*



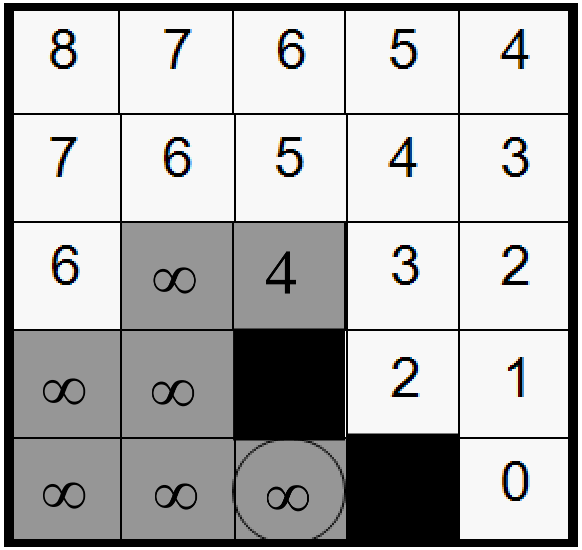
update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s') + h(s'))$
until convergence

\- *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 = argmin* $_{s' \in OPEN}$ *g(s') + h(s')*



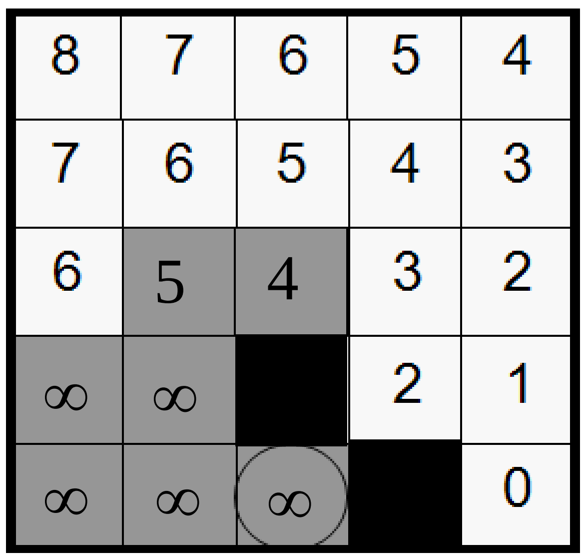
update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s')+h(s'))$
until convergence

- *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* $s = \text{argmin}_{s' \in OPEN}\, g(s') + h(s')$



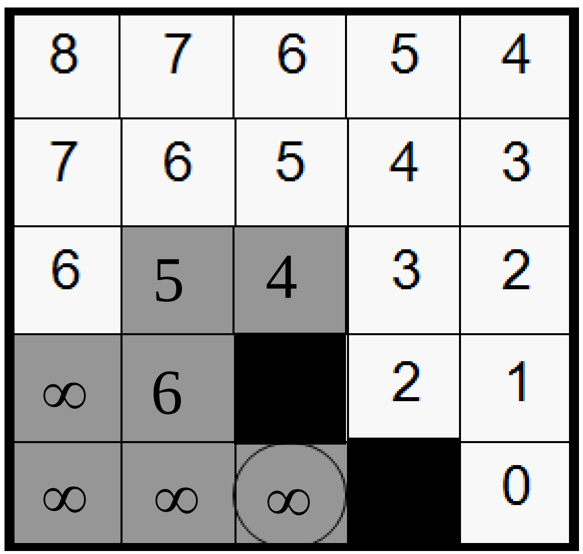
update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s')+h(s'))$
until convergence

- *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 = argmin* $_{s' \in OPEN}\, g(s') + h(s')$



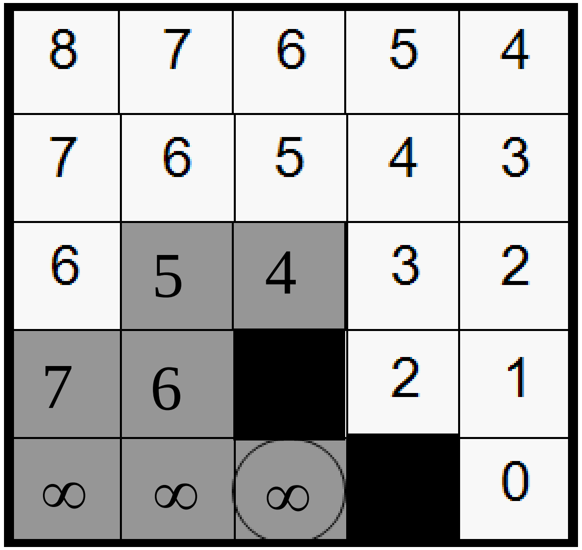
update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s') + h(s'))$
until convergence

- *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 = argmin* $_{s' \in OPEN}$ *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

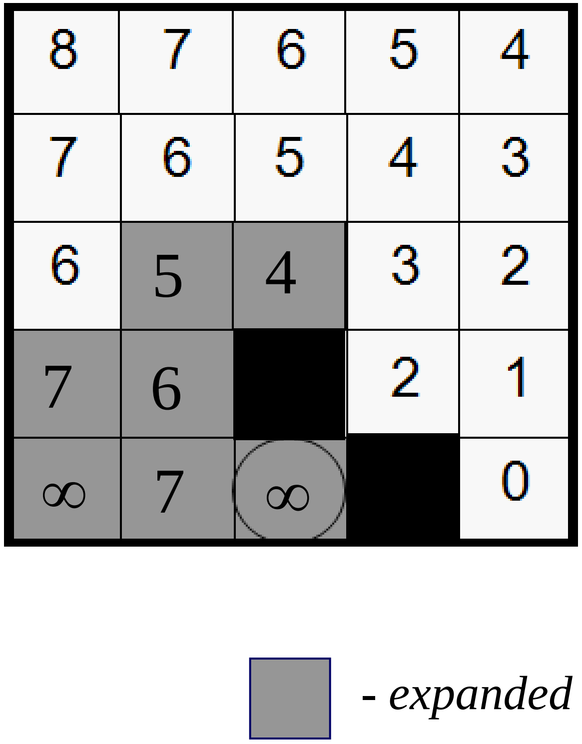
Does it matter in what order?

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

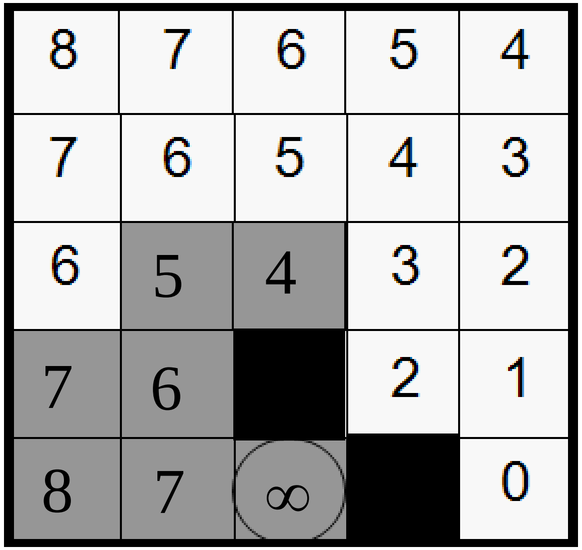


update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s') + h(s'))$
until convergence

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 = argmin* $_{s' \in OPEN}\, g(s') + h(s')$



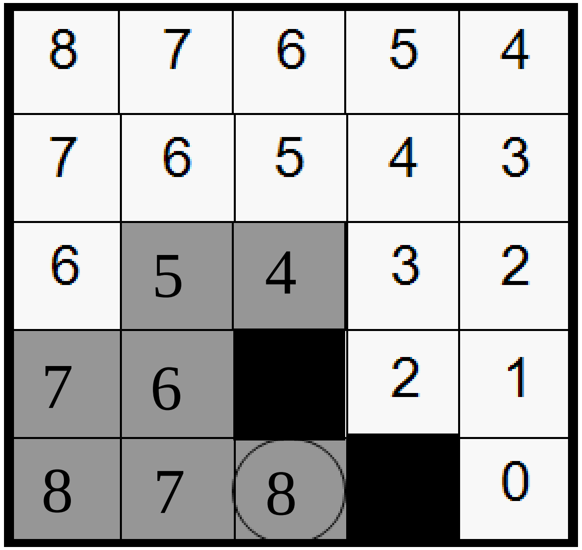
update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s')+h(s'))$
until convergence

- *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* $s = \text{argmin}_{s' \in OPEN}\, g(s') + h(s')$



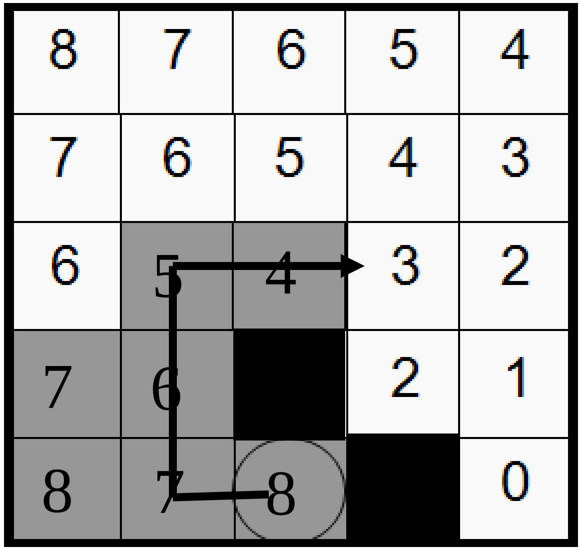
update h-values of expanded states via DP:
compute $h(s) = \min_{s' \in succ(s)} (c(s,s') + h(s'))$
until convergence


- *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* $s = \text{argmin}_{s' \in OPEN}\, g(s') + h(s')$



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

Drawbacks compared to A?*

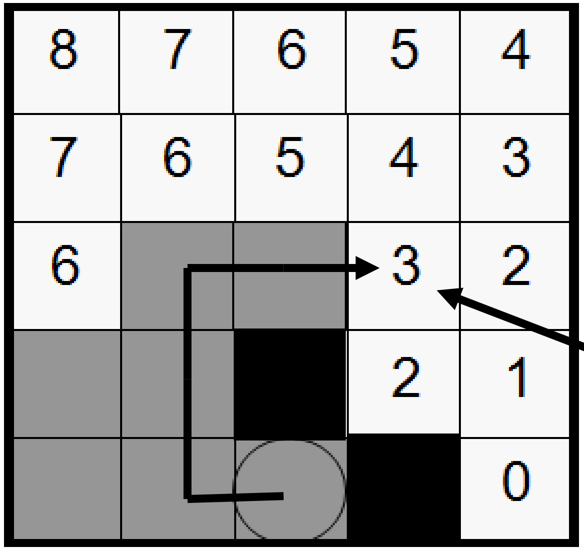
- expanded

Real-time Adaptive A* (RTAA*)

- RTAA* with $N \geq 1$ expands

one linear pass
(and even that can be postponed)

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



expand N=7 states

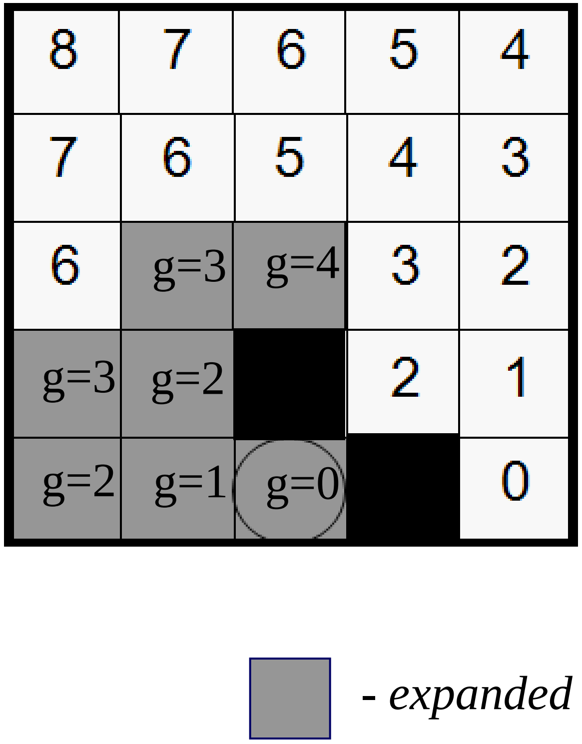
unexpanded state *s* with smallest *g* + *h* (= 5 + 3)

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



update all expanded states u:

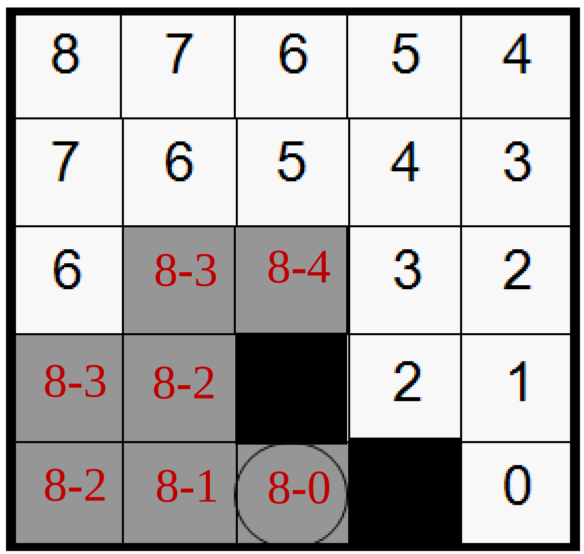
$h(u) = f(s) - g(u)$

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



update all expanded states u:

h(u) = f(s) – g(u)

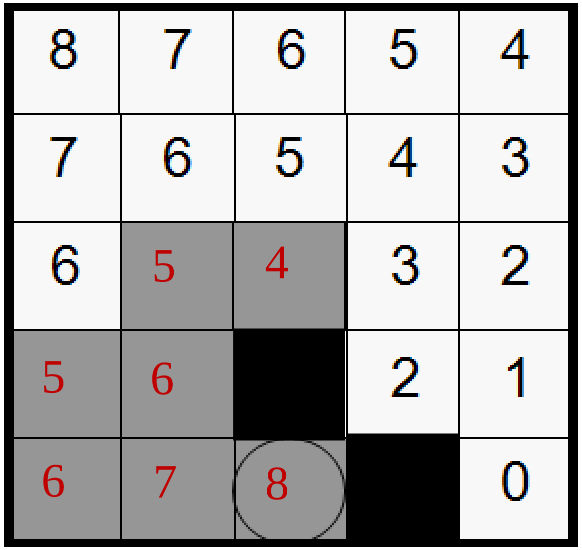
unexpanded state *s* with smallest *f(s) = 8*

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



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

proof of admissibility:

gd() – true cost-to-goal

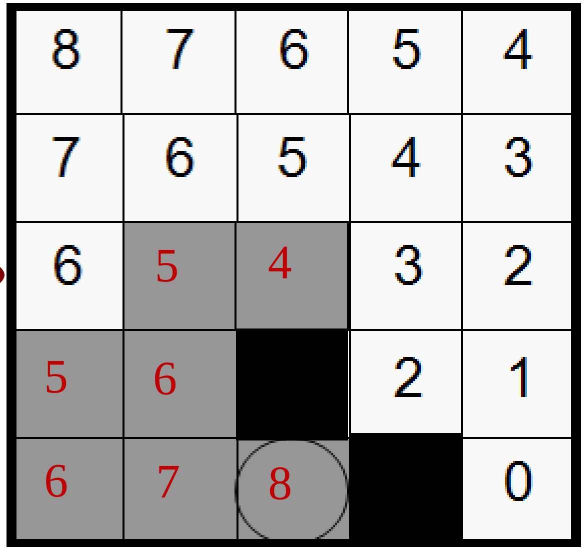
$$g(u) + gd(u) \geq gd(s_{start})$$

$$gd(u) \geq gd(s_{start}) - g(u)$$

because $f(s) \leq gd(s_{start})$

$$gd(u) \geq f(s) - g(u)$$

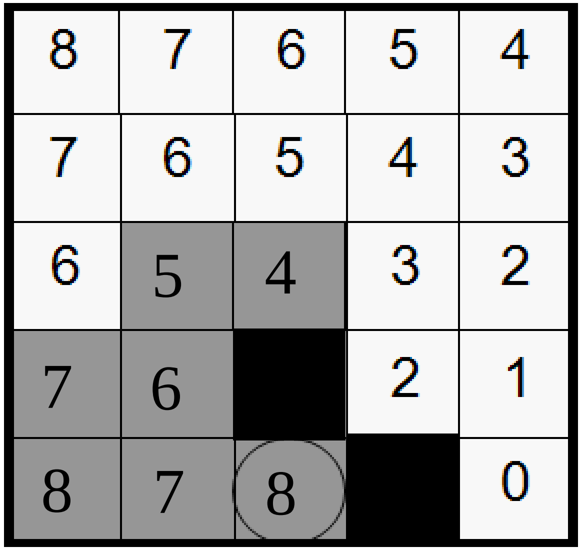
$$gd(u) \geq h_{updated}(u)$$



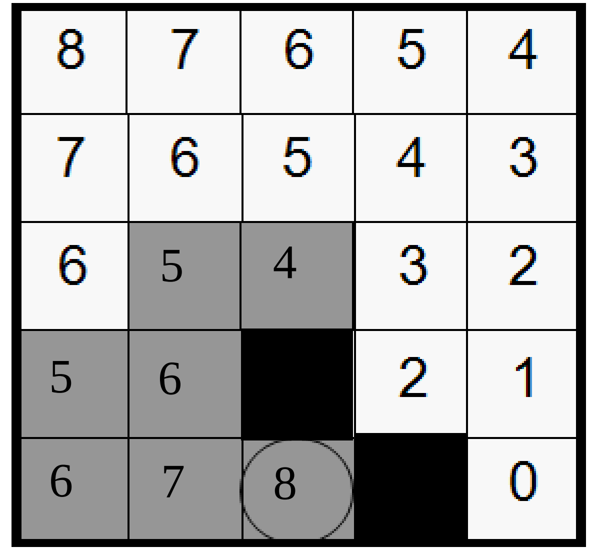
- *expanded*

LRTA* vs. RTAA*

LRTA*



RTAA*



- Update of *h*-values in RTAA* is much faster but not as informed
- Both guarantee adimssibility 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)