

***16-782***

***Planning & Decision-making in Robotics***

***Search Algorithms:  
Heuristic Functions, Multi-Heuristic A\****

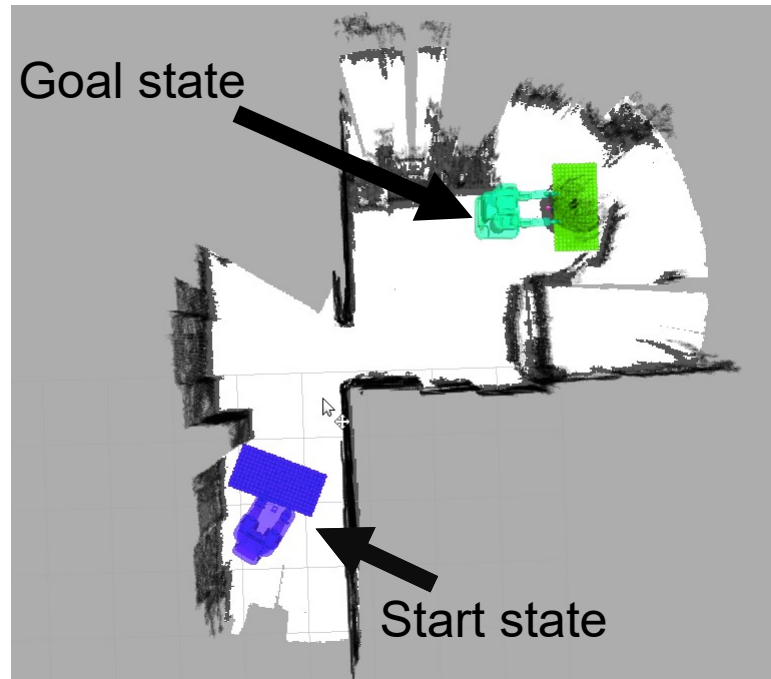
*Maxim Likhachev*

*Robotics Institute*

*Carnegie Mellon University*

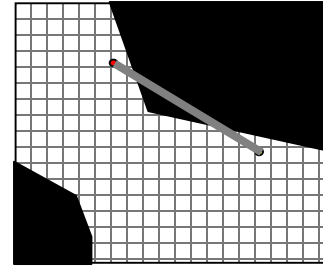
# Design of Informative Heuristics

Example problem: *move picture frame on the table*



- Full-body planning
- 12 Dimensions  
(3D base pose,  
1D torso height,  
6DOF object pose,  
2 redundant DOFs  
in arms)

# Design of Informative Heuristics



- For grid-based navigation:
  - Euclidean distance
  - Manhattan distance:  $h(x,y) = \text{abs}(x-x_{goal}) + \text{abs}(y-y_{goal})$
  - Diagonal distance:  $h(x,y) = \max(\text{abs}(x-x_{goal}), \text{abs}(y-y_{goal}))$
  - More informed distances???

*Which heuristics are admissible for  
4-connected grid?  
8-connected grid?*

# Design of Informative Heuristics

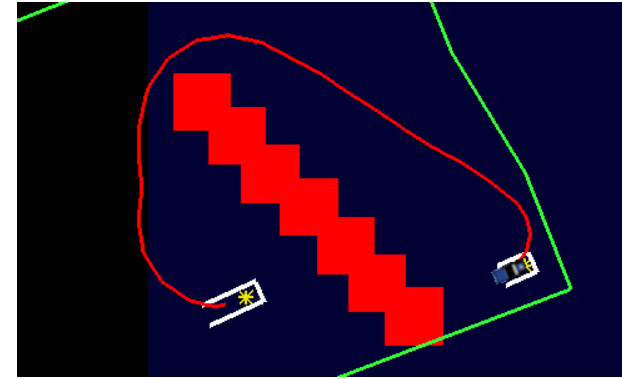
- For lattice-based 3D  $(x, y, \theta)$  navigation:

*Any ideas?*



# Design of Informative Heuristics

- For lattice-based 3D  $(x, y, \theta)$  navigation:



- 2D  $(x, y)$  distance accounting for obstacles (single Dijkstra's on 2D grid cell starting at goalcell will give us these values)

# Design of Informative Heuristics



- For lattice-based 3D  $(x, y, \theta)$  navigation:
  - 2D  $(x, y)$  distance accounting for obstacles (single Dijkstra's on 2D grid cell starting at goalcell will give us these values)

*Any problems where it will be highly uninformative?*

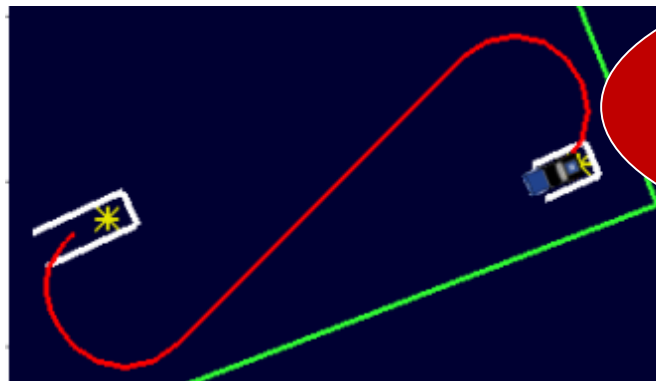
# Design of Informative Heuristics

- For lattice-based 3D  $(x,y,\Theta)$  navigation:



- 2D  $(x,y)$  distance accounting for obstacles (single Dijkstra's on 2D grid cell starting at goalcell will give us these values)

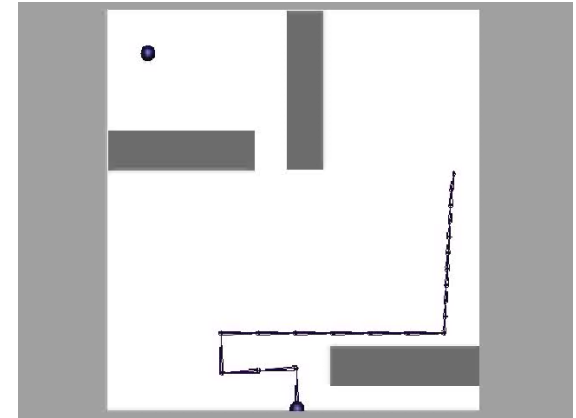
*Any problems where it will be highly uninformative?*



*Any heuristic functions that will guide search well in this example?*

# Design of Informative Heuristics

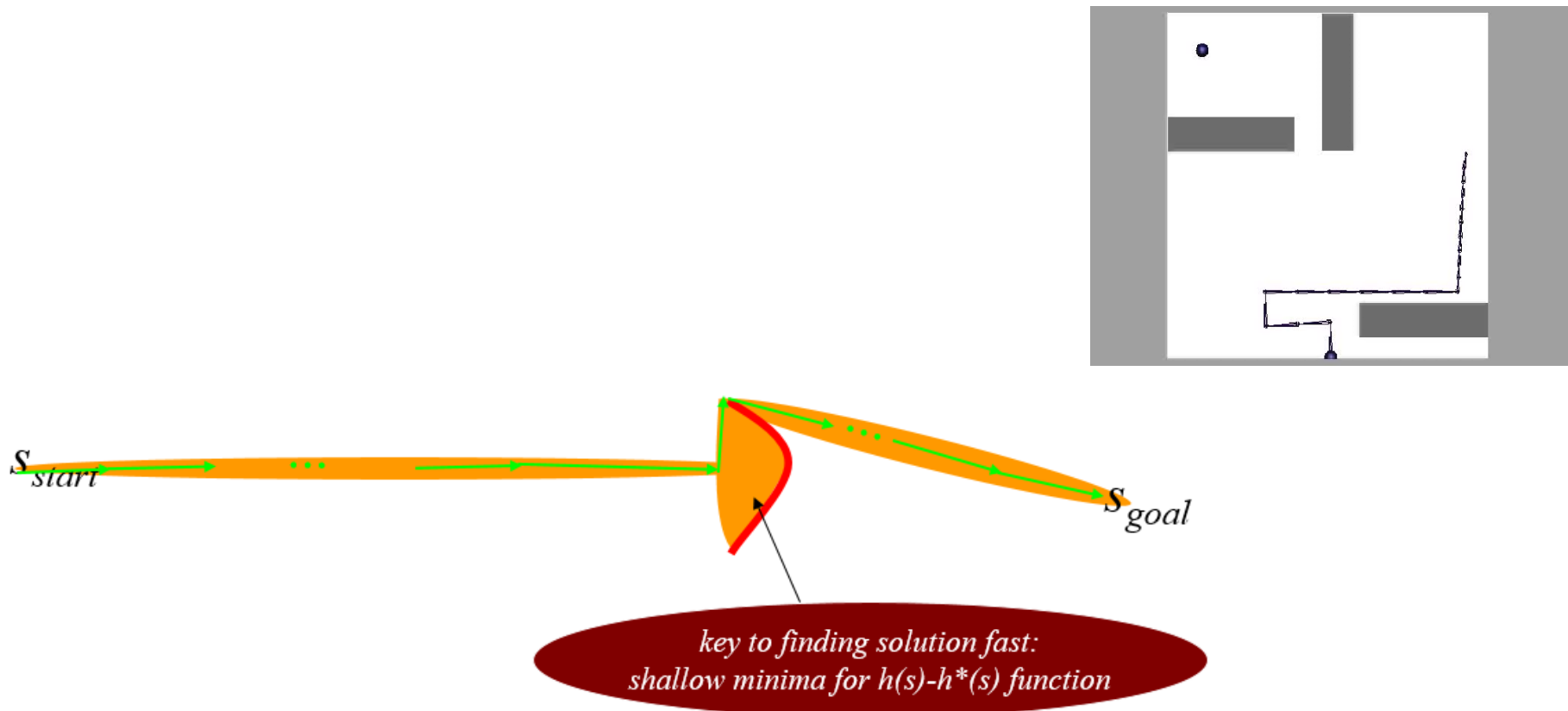
- 20DoF Planar arm planning (*forget optimal  $A^*$ , use weighted  $A^*$* ):





# Design of Informative Heuristics

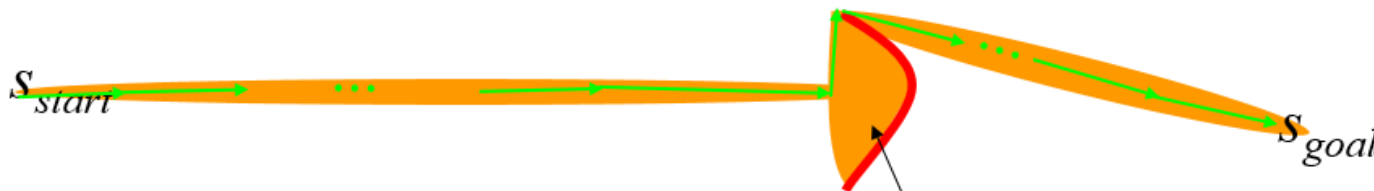
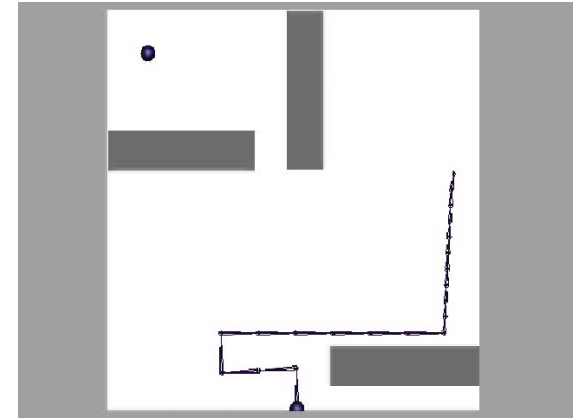
- 20DoF Planar arm planning (*forget optimal  $A^*$ , use weighted  $A^*$* ):



# Design of Informative Heuristics

- 20DoF Planar arm planning (*forget optimal  $A^*$ , use weighted  $A^*$* ):

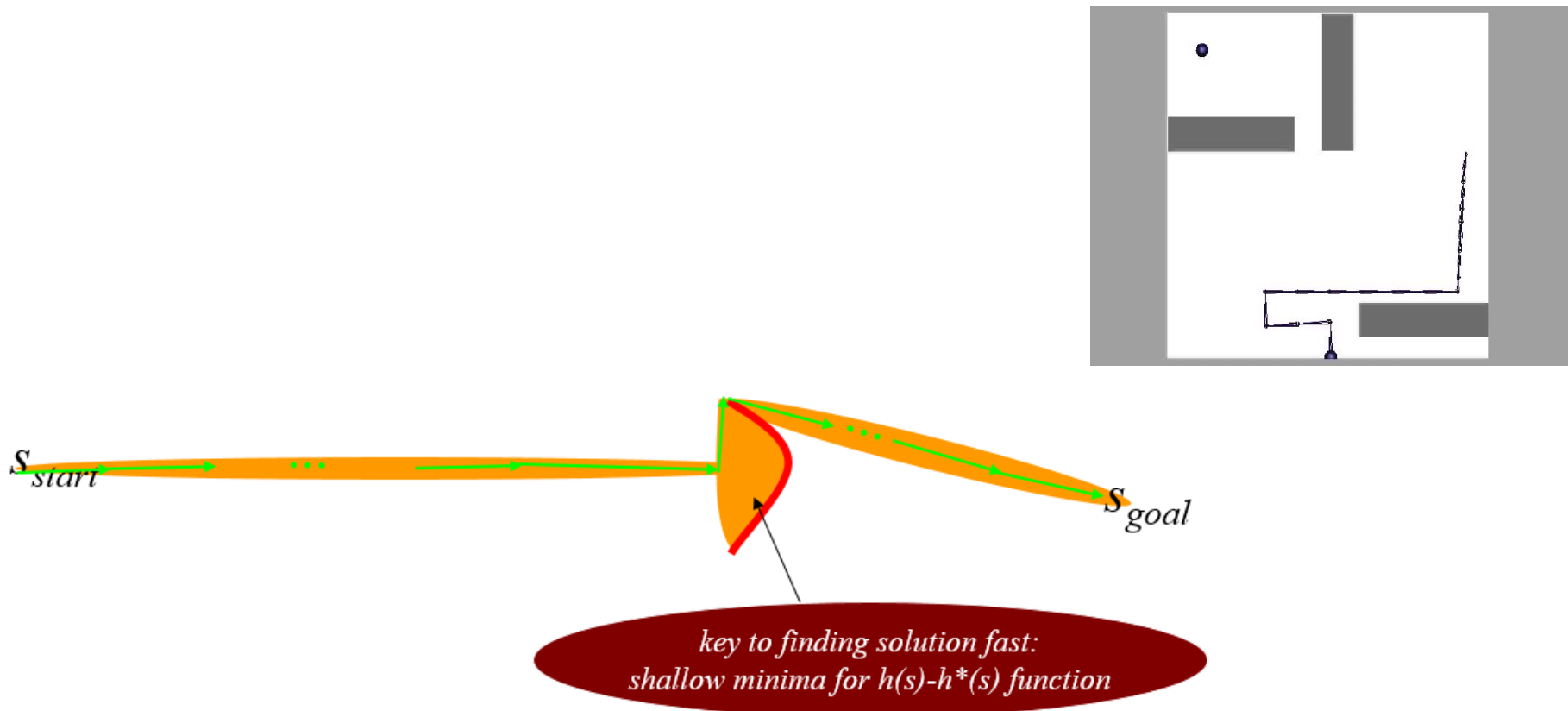
*Any ideas?*



*key to finding solution fast:  
shallow minima for  $h(s)-h^*(s)$  function*

# Design of Informative Heuristics

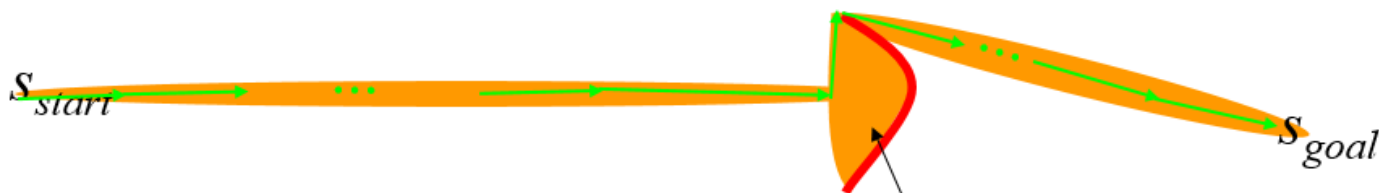
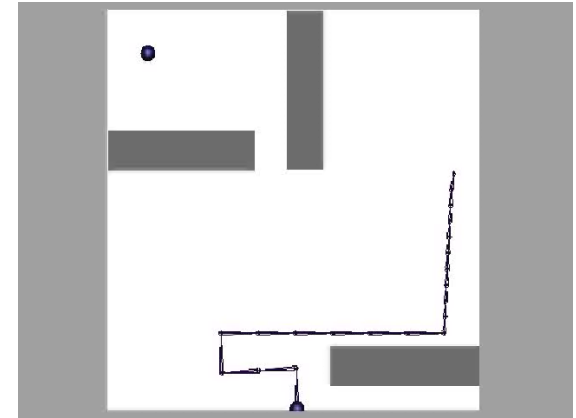
- 20DoF Planar arm planning (*forget optimal  $A^*$ , use weighted  $A^*$* ):
  - 2D end-effector distance accounting for obstacles



# Design of Informative Heuristics

- 20DoF Planar arm planning (*forget optimal  $A^*$ , use weighted  $A^*$* ):
  - 2D end-effector distance accounting for obstacles

*Example where it will miserably fail?*

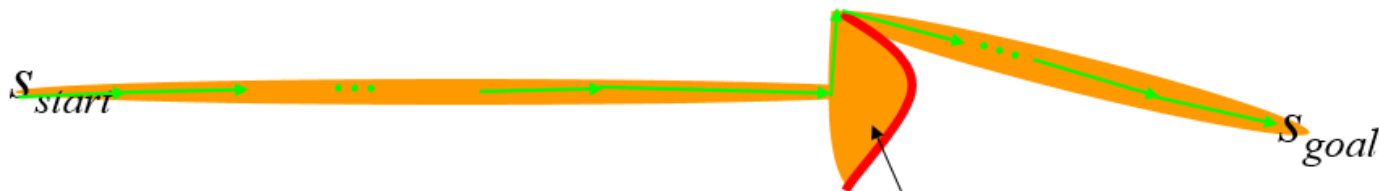
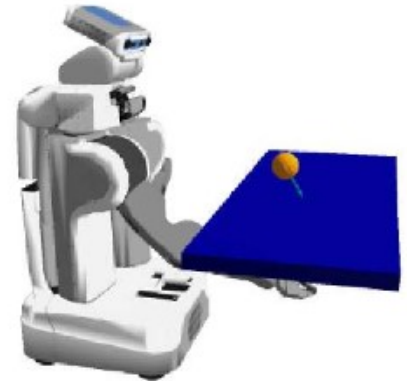


key to finding solution fast:  
shallow minima for  $h(s)-h^*(s)$  function

# Design of Informative Heuristics

- Arm planning in 3D:

*Any ideas?*

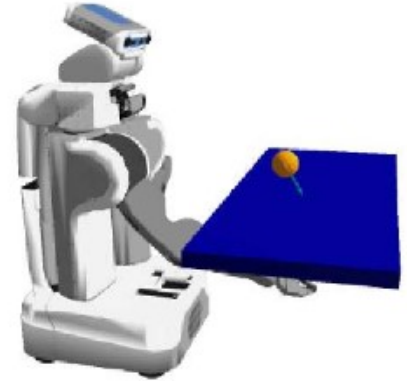


*key to finding solution fast:  
shallow minima for  $h(s)-h^*(s)$  function*

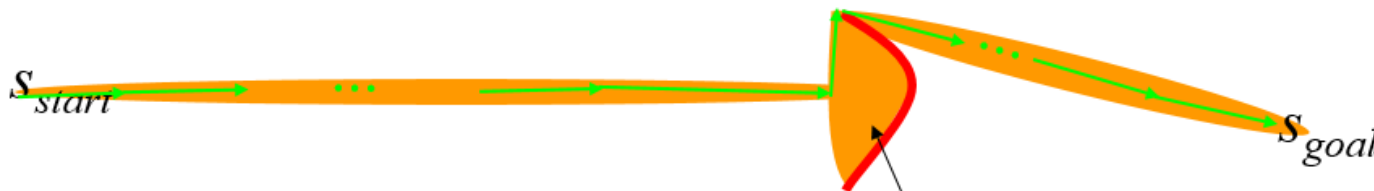
# Design of Informative Heuristics

- Arm planning in 3D:

*Any ideas?*



- 3D  $(x,y,z)$  end-effector distance accounting for obstacles



*key to finding solution fast:  
shallow minima for  $h(s)-h^*(s)$  function*

# Few Properties of Heuristic Functions

- Useful properties to know:

- $h_1(s), h_2(s)$  – consistent, then:

$$h(s) = \max(h_1(s), h_2(s)) \text{ – consistent}$$

- if  $A^*$  uses  $\varepsilon$ -consistent heuristics:

$$h(s_{goal}) = 0 \text{ and } h(s) \leq \varepsilon c(s, succ(s)) + h(succ(s)) \text{ for all } s \neq s_{goal},$$

then  $A^*$  is  $\varepsilon$ -suboptimal:

$$cost(solution) \leq \varepsilon cost(optimal\ solution)$$

- weighted  $A^*$  is  $A^*$  with  $\varepsilon$ -consistent heuristics

- $h_1(s), h_2(s)$  – consistent, then:

$$h(s) = h_1(s) + h_2(s) \text{ – } \varepsilon\text{-consistent}$$

# Few Properties of Heuristic Functions

- Useful properties to know:

- $h_1(s), h_2(s)$  – consistent, then:

$$h(s) = \max(h_1(s), h_2(s)) \text{ – consistent}$$

- if  $A^*$  uses  $\varepsilon$ -consistent heuristics:

$$h(s_{goal}) = 0 \text{ and } h(s) \leq \varepsilon c(s, succ(s)) + h(succ(s)) \text{ for all } s \neq s_{goal},$$

then  $A^*$  is  $\varepsilon$ -suboptimal:

$$cost(solution) \leq \varepsilon cost(optimal\ solution)$$

- weighted  $A^*$  is  $A^*$  with  $\varepsilon$ -consistent heuristics

*Proof?*

- $h_1(s), h_2(s)$  – consistent, then:

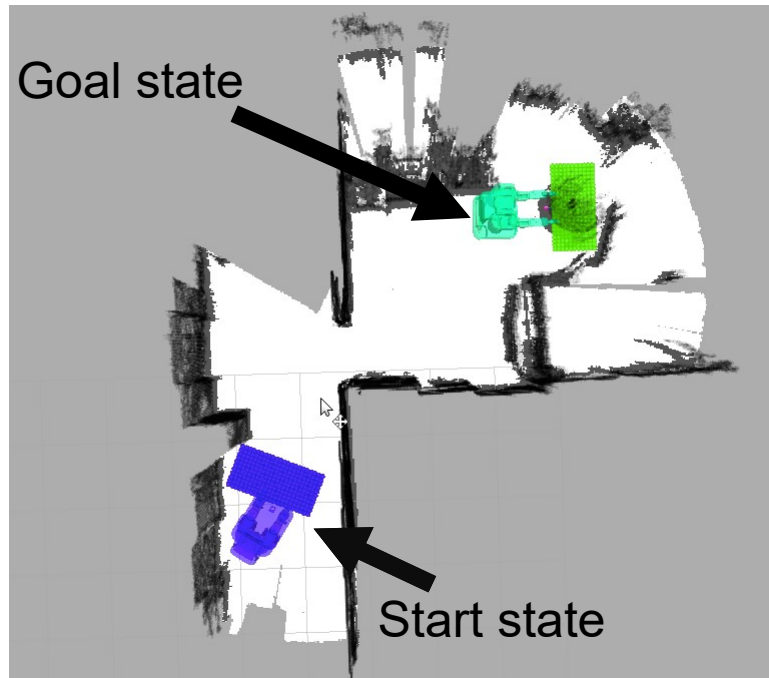
$$h(s) = h_1(s) + h_2(s) \text{ – } \varepsilon\text{-consistent}$$

*What is  $\varepsilon$ ? Proof?*



# Back to the Example

Example problem: *move picture frame on the table*

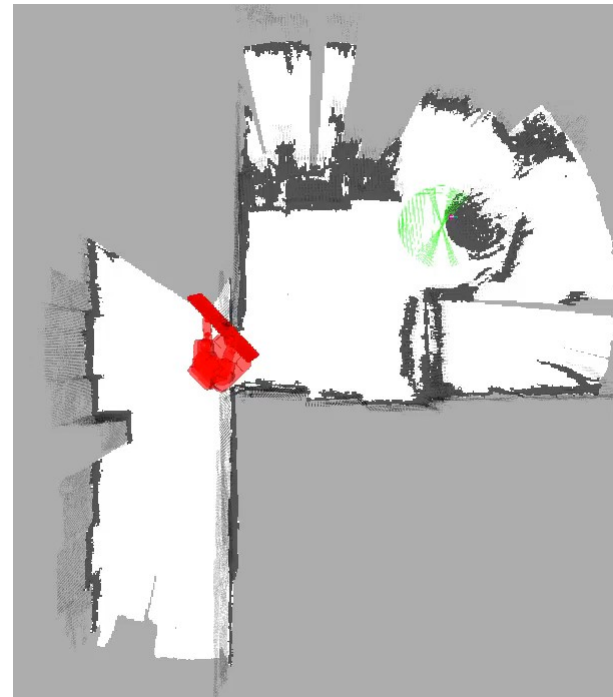
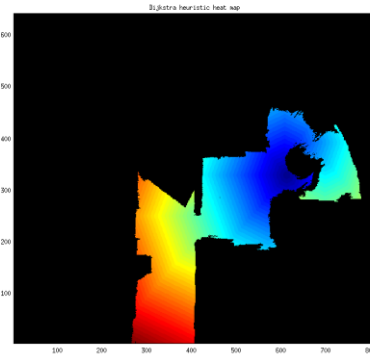
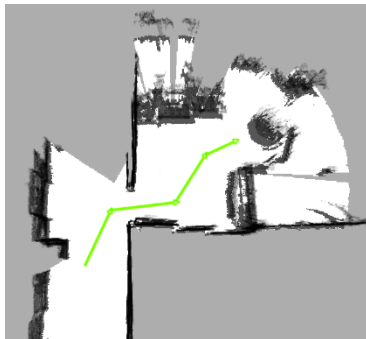


- Full-body planning
- 12 Dimensions  
(3D base pose,  
1D torso height,  
6DOF object pose,  
2 redundant DOFs  
in arms)

# Back to the Example

## Admissible and Consistent Heuristic

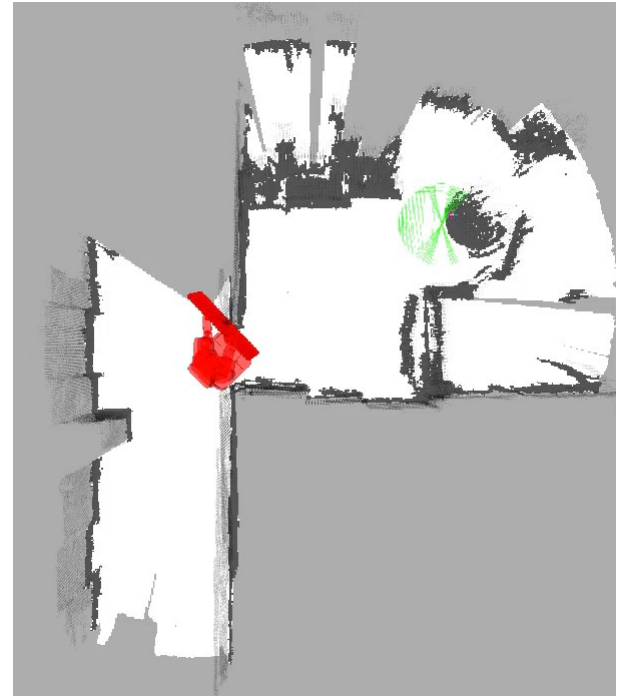
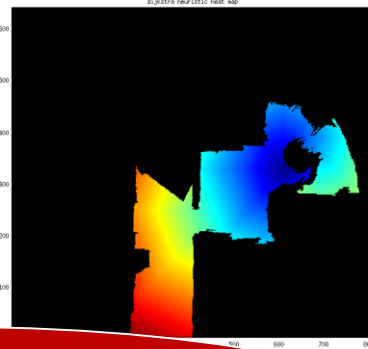
- $h_o$ : base distance
  - 2D BFS from goal state



# Back to the Example

## Admissible and Consistent Heuristic

- $h_0$ : base distance
  - 2D BFS from goal state



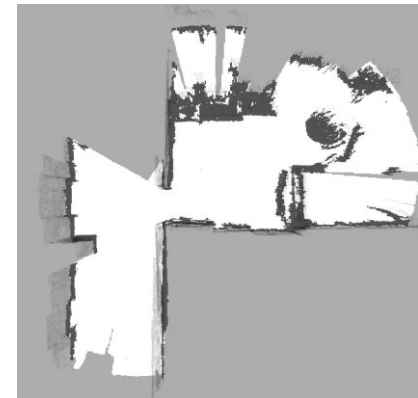
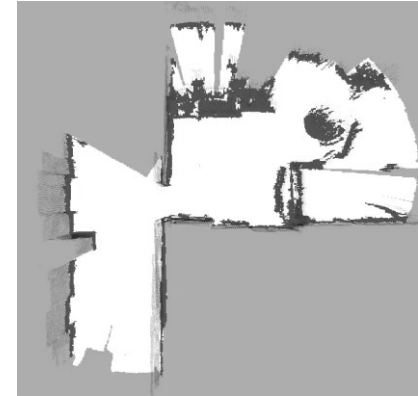
Do you think it will guide search well?

Any other ideas for good heuristics?

# Back to the Example

## Inadmissible Heuristics

- $h_1$ : base distance + object orientation difference with goal
- $h_2$ : base distance + object orientation difference with vertical



# Back to the Example

*More generally: we can often easily generate  $N$  arbitrary heuristic functions that estimate costs-to-goal*

*Solutions to  $N$  lower-dimensional manifolds  
Solutions to  $N$  problems with different constraints relaxed*

....

- $h_2$ : base distance + object orientation difference with vertical

# Utilizing Multiple Heuristic Functions

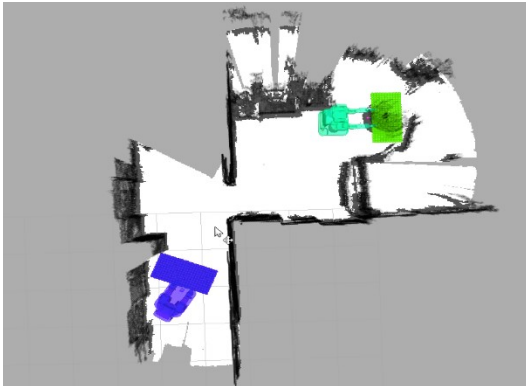
---

***Can we utilize a bunch of inadmissible heuristics simultaneously, leveraging their individual strengths while preserving guarantees on completeness and bounded sub-optimality?***

# Utilizing Multiple Heuristic Functions

*Can we utilize a bunch of inadmissible heuristics simultaneously, leveraging their individual strengths while preserving guarantees on completeness and bounded sub-optimality?*

*Combining multiple heuristics into one (e.g., taking max) is often inadequate*



- *information is lost*
- *creates local minima*
- *requires all heuristics to be admissible*

# Multi-Heuristics A\*: version 1

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal

Within the while loop of the ComputePath function:

*for  $i=1 \dots N$*

*remove  $s$  with the smallest  $[f(s) = g(s) + w_1 * h(s)]$  from  $OPEN_i$ ;*

*expand  $s$ ;*

Inad. Search 1

Inad. Search 2

Inad. Search 3

priority queue:  $OPEN_1$   
key =  $g + w_1 * h_1$

priority queue:  $OPEN_2$   
key =  $g + w_1 * h_2$

priority queue:  $OPEN_3$   
key =  $g + w_1 * h_3$



# Multi-Heuristics A\*: version 1

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal

Problems:

- Each search has its own local minima
- N times more work
- No completeness guarantees or bounds on solution quality

Inad. Search 1

priority queue:  $OPEN_1$   
key =  $g + w_1 * h_1$

Inad. Search 2

priority queue:  $OPEN_2$   
key =  $g + w_1 * h_2$

Inad. Search 3

priority queue:  $OPEN_3$   
key =  $g + w_1 * h_3$

# Multi-Heuristics A\*: version 2

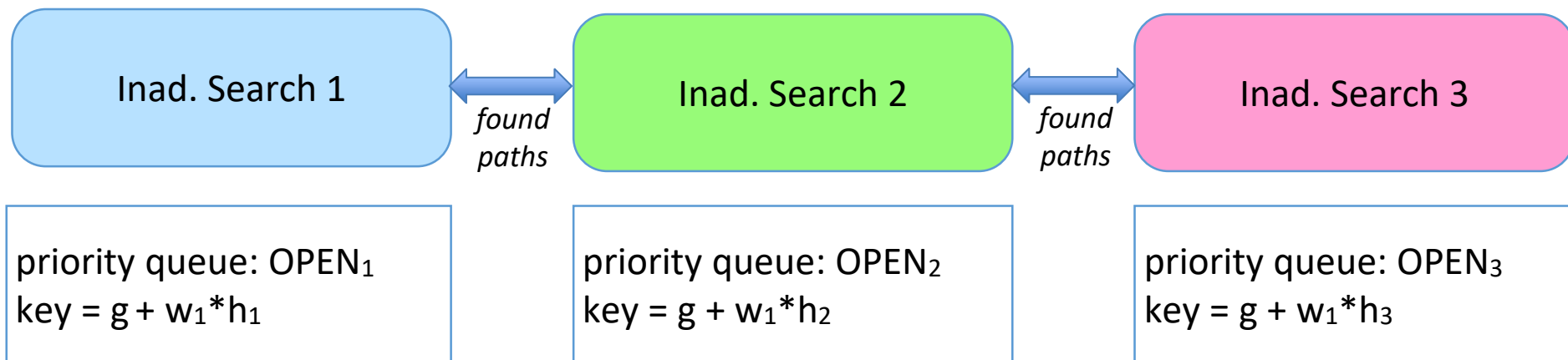
- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- **Key Idea #1: Share information (g-values) between searches!**

Within the while loop of the ComputePath function (note: CLOSED is shared):

*for  $i=1 \dots N$*

*remove  $s$  with the smallest  $[f(s) = g(s) + w_1 * h(s)]$  from  $OPEN_i$ ;*

*expand  $s$  **and also insert/update its successors into all other OPEN lists**;*



# Multi-Heuristics A\*: version 2

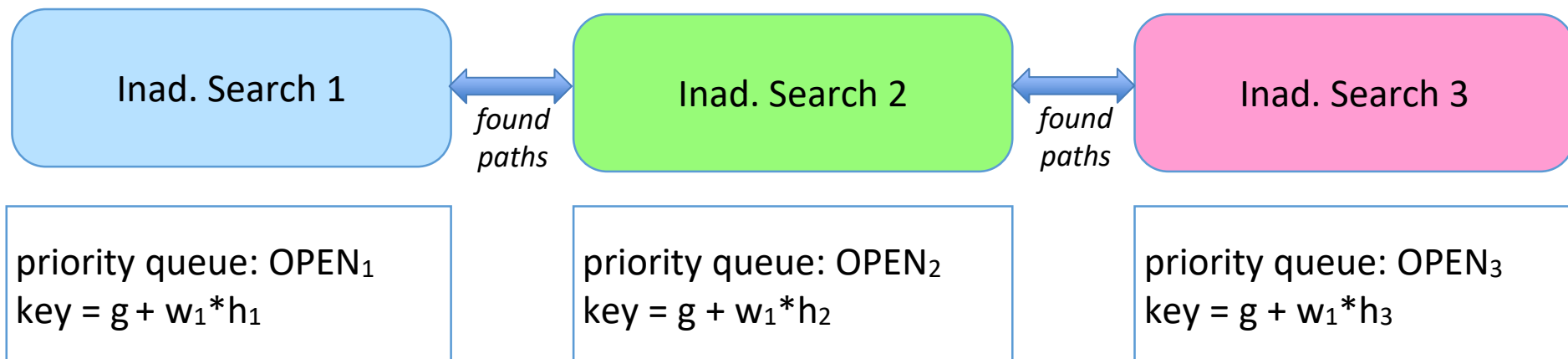
- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- **Key Idea #1: Share information (g-values) between searches!**

Benefits:

- Searches help each other to circumvent local minima
- States are expanded at most once across ALL searches

Remaining Problem:

- No completeness guarantees or bounds on solution quality

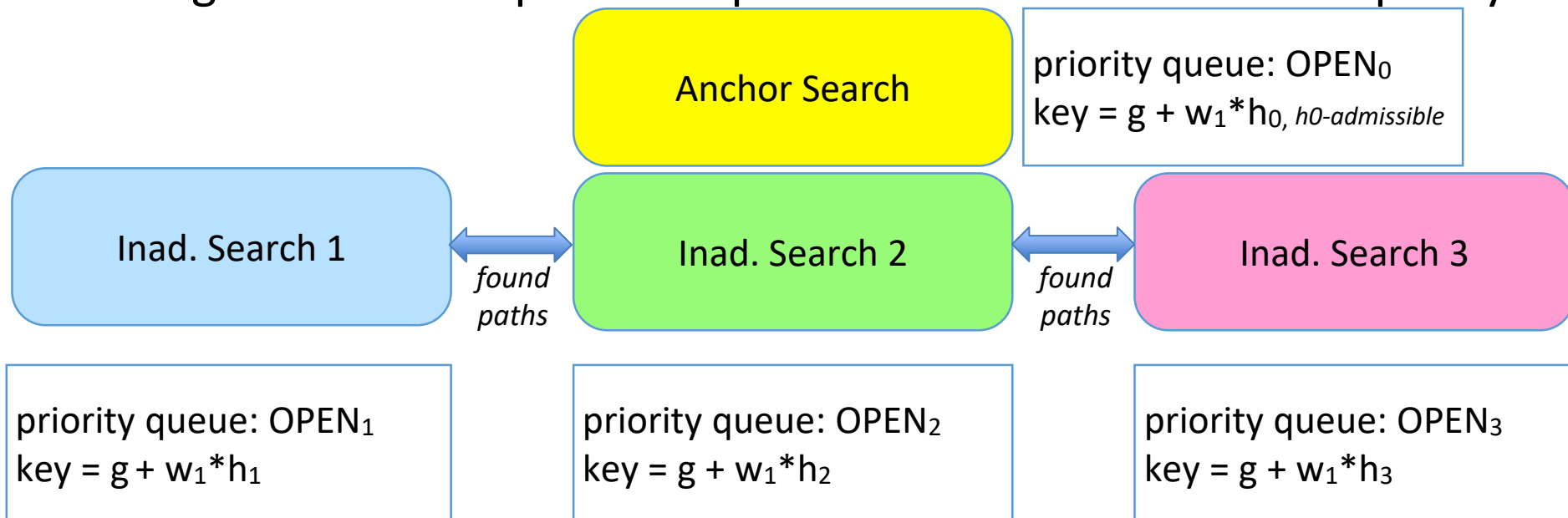


# Multi-Heuristics A\* [Aine et al., '14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- **Key Idea #1: Share information (g-values) between searches!**
- **Key Idea #2: Search with admissible heuristics controls expansions**

Benefits:

- Algorithm is complete and provides bounds on solution quality

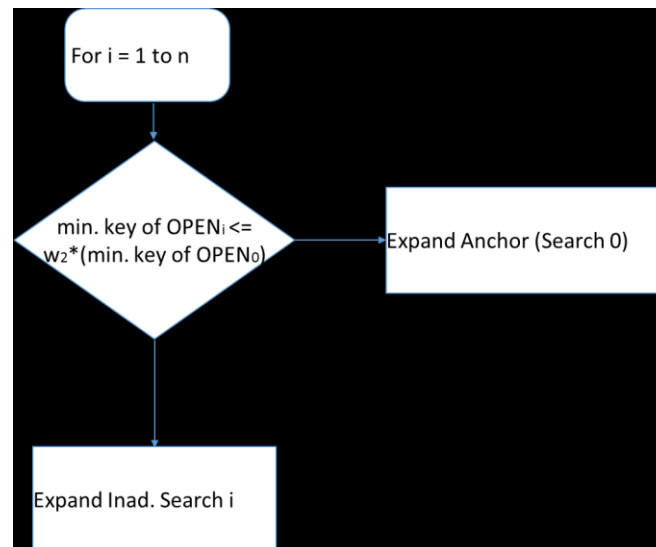


# Multi-Heuristics A\* [Aine et al., '14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- **Key Idea #1: Share information (g-values) between searches!**
- **Key Idea #2: Search with admissible heuristics controls expansions**

Benefits:

- Algorithm is complete and provides bounds on solution quality



# Multi-Heuristics A\* [Aine et al., '14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- **Key Idea #1: Share information (g-values) between searches!**
- **Key Idea #2: Search with admissible heuristics controls expansions**

Benefits:

- Algorithm is complete and provides bounds on solution quality

Within the while loop of the ComputePath function

(note: CLOSED is shared among searches 1...N. Search 0 has its own CLOSED):

*for  $i=1 \dots N$*

*if ( $\min. f\text{-value in } OPEN_i \leq w_2 * \min. f\text{-value in } OPEN_0$ )*

*remove  $s$  with the smallest  $[f(s) = g(s) + w_1 * h_i(s)]$  from  $OPEN_i$ ;*

*expand  $s$  and also insert/update its successors into all other OPEN lists;*

*else*

*remove  $s$  with the smallest  $[f(s) = g(s) + w_1 * h_0(s)]$  from  $OPEN_0$ ;*

*expand  $s$  and also insert/update its successors into all other OPEN lists;*

# Multi-Heuristics A\* [Aine et al., '14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them is optimal

Theorem 1: *min. key of  $OPEN_0 \leq w_1 * \text{optimal solution cost}$*

- **Key Idea #1: Share information / combine results**

- **Key Idea #2: Search**

Theorem 2: *min. key of  $OPEN_i \leq w_2 * w_1 * \text{optimal solution cost}$*

Benefits:

- Algorithm is complete

Theorem 3: *The algorithm is complete  
and the cost of the found solution is no more than  
 $w_2 * w_1 * \text{optimal solution cost}$*

Within the while loop of the algorithm

(note: CLOSED is shared across all searches)

for  $i=1 \dots N$

Theorem 4: *Each state is expanded at most twice:  
at most once by one of the inadmissible searches  
and at most once by the Anchor search*

*if (min.  $f$ -value in  $OPEN_i \leq w_2 * \text{min. } f\text{-value in } OPEN_0$ )*

*remove  $s$  with the smallest  $[f(s) = g(s) + w_1 * h_i(s)]$  from  $OPEN_i$ ;*

*expand  $s$  and also insert/update its successors into all other OPEN lists;*

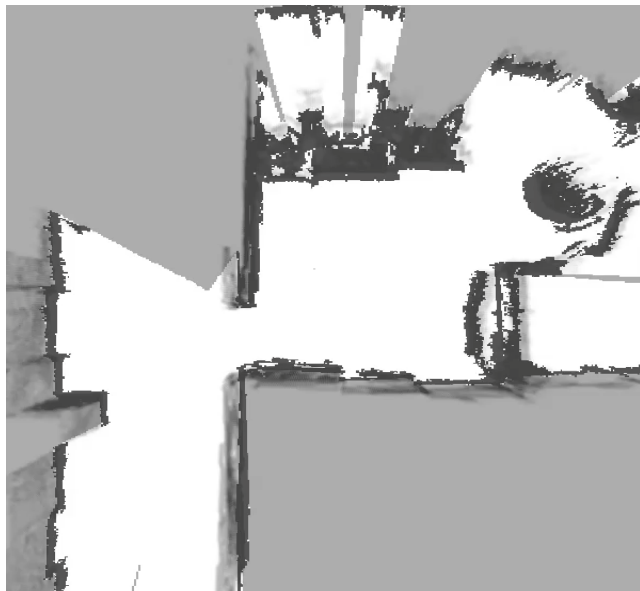
*else*

*remove  $s$  with the smallest  $[f(s) = g(s) + w_1 * h_0(s)]$  from  $OPEN_0$ ;*

*expand  $s$  and also insert/update its successors into all other OPEN lists;*

# Multi-Heuristics A\* [Aine et al., '14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- **Key Idea #1: Share information (g-values) between searches!**
- **Key Idea #2 Search with admissible heuristics controls expansions**





# What You Should Know...

- Examples of heuristic functions
  - for X-connected grids
  - For higher dimensional planning problems derived by lower-dimensional search
- Be able to come up with a good heuristic function for a given problem
- Properties of heuristic functions
- How Multi-heuristic  $A^*$  works