Planning Algorithms: When Optimal Just Isn't Good Enough

Wheeler Ruml

University of New Hampshire
Department of Computer Science

Joint work with the UNH AI Group. Funded by NSF and DARPA.

Where are the Walking Talking Robots?

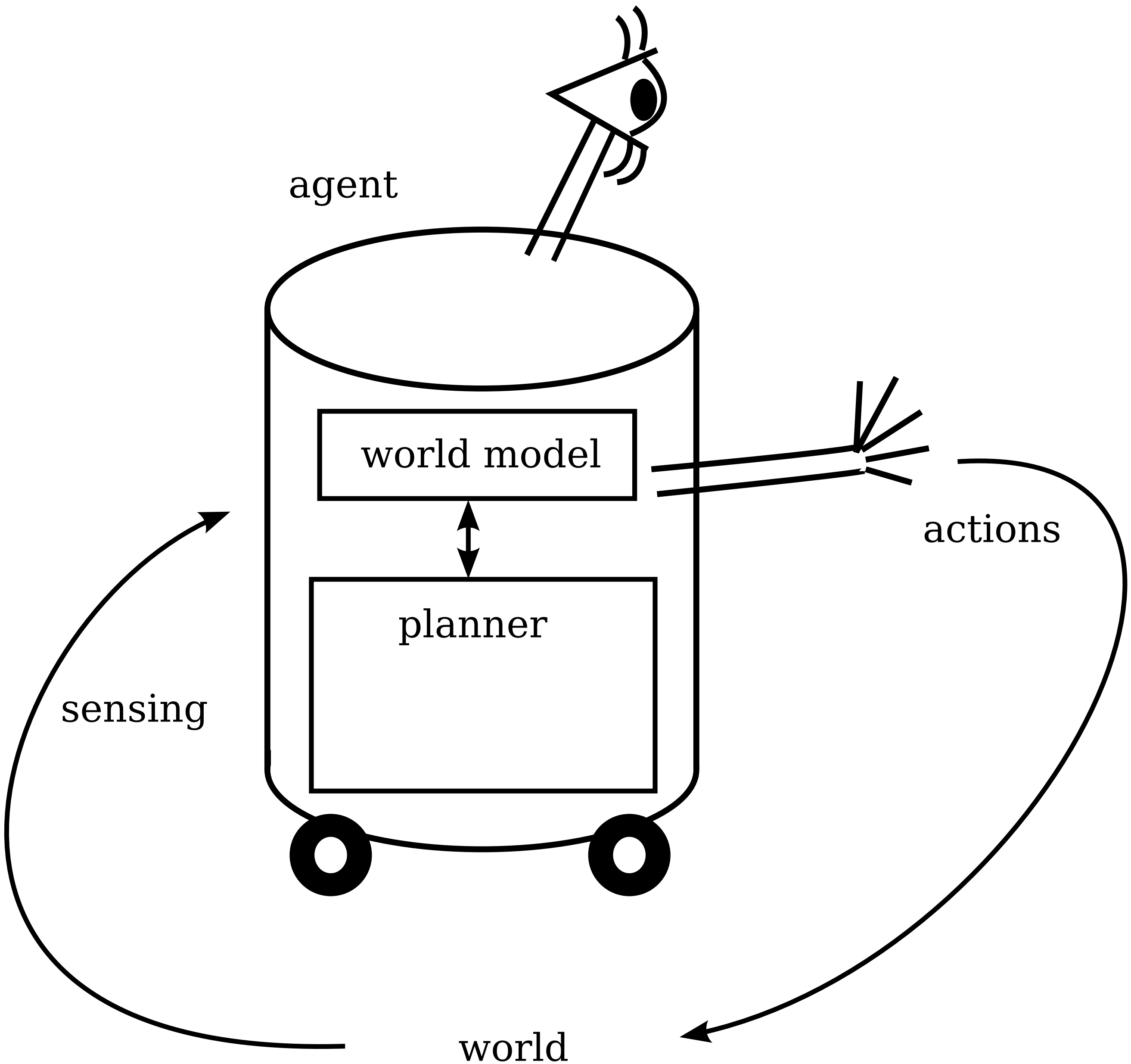
Lt. Commander Data
Star Trek: TNG

Lt. Sharon Valerii
Battlestar Galactica

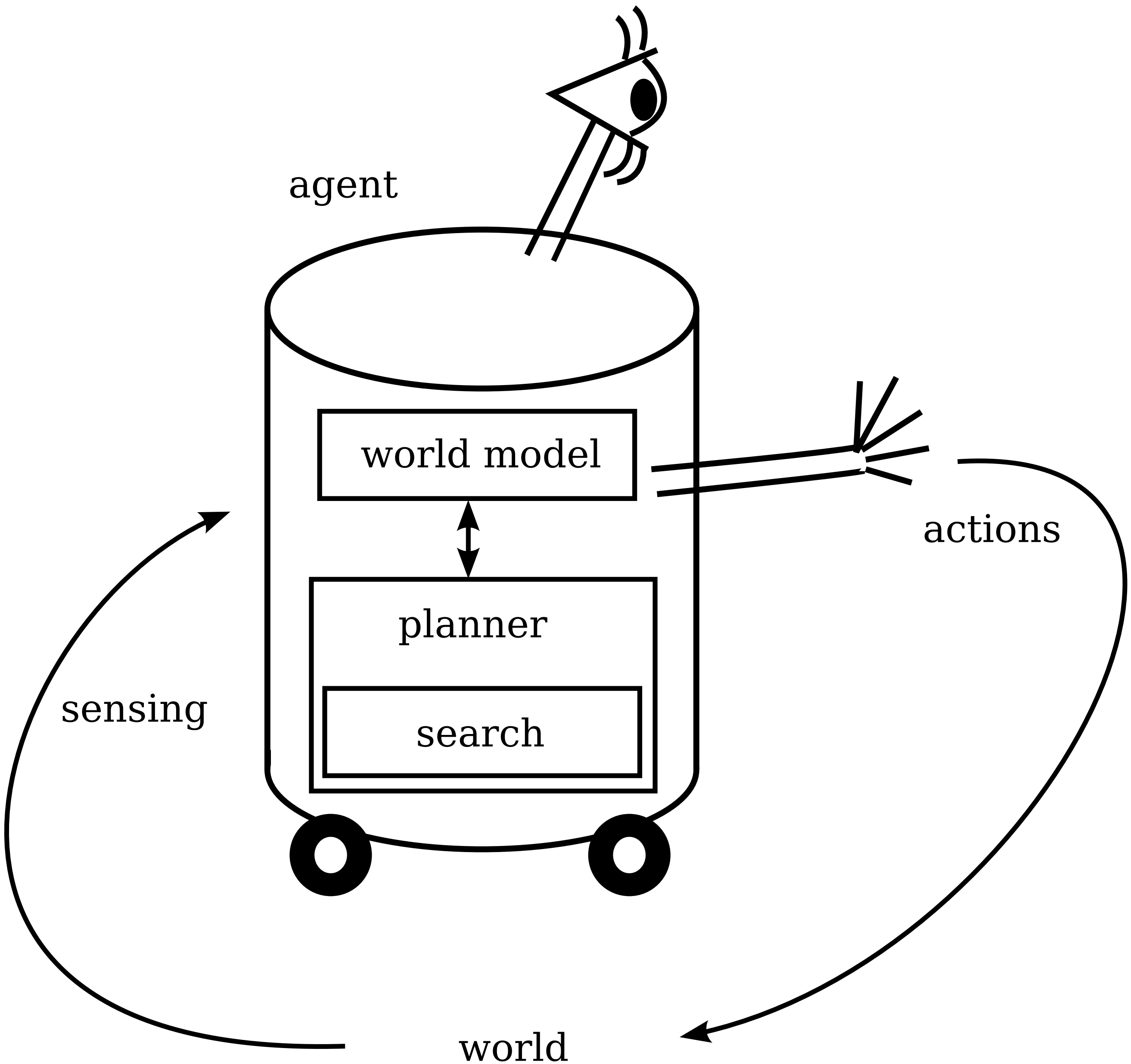
Zoe Greystone
Caprica

Toddler Steps

The Role of Planning



The Role of Planning



Planning as Heuristic Graph Search

Planning

Given:

- current state of the world
- models of available actions
  preconditions, effects, costs
- desired state of the world (partially specified?)

Find:

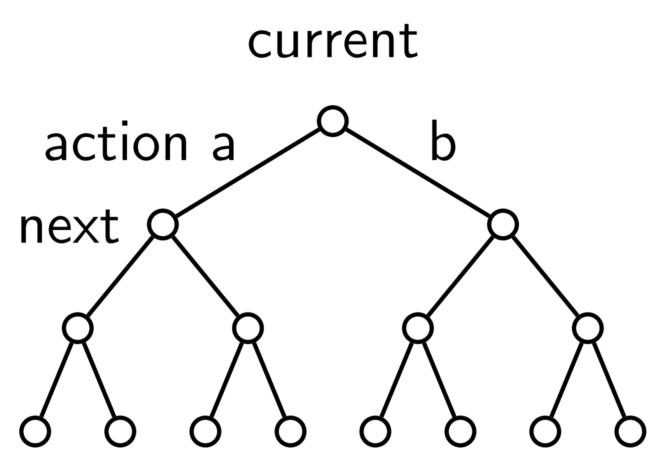
- cheapest plan

Given:

- start state: an explicit node
- expand function: generates children and their costs
  assume arc cost ≥ 0
- goal test: predicate on nodes

Find:

- cheapest path to a goal node



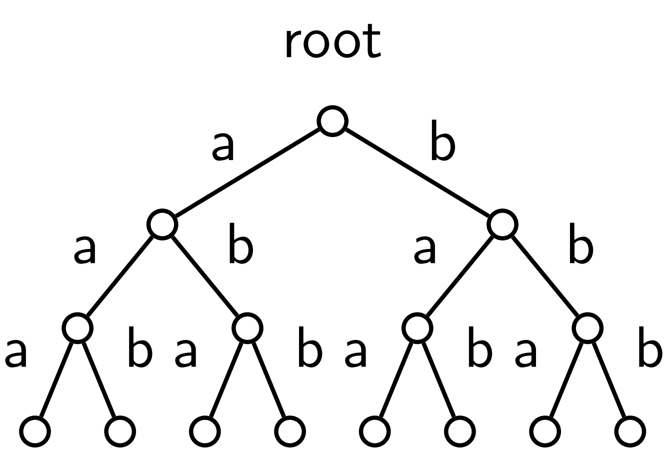
Graph Search

Given:

- start state: an explicit node
- expand function: generates children and their costs
  assume arc cost ≥ 0
- goal test: predicate on nodes

Find:

- cheapest path to a goal node



Graph Search

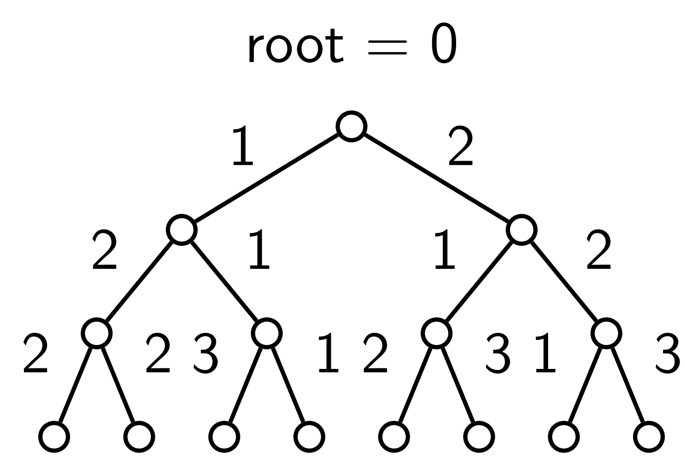
Given:

- start state: an explicit node
- expand function: generates children and their costs

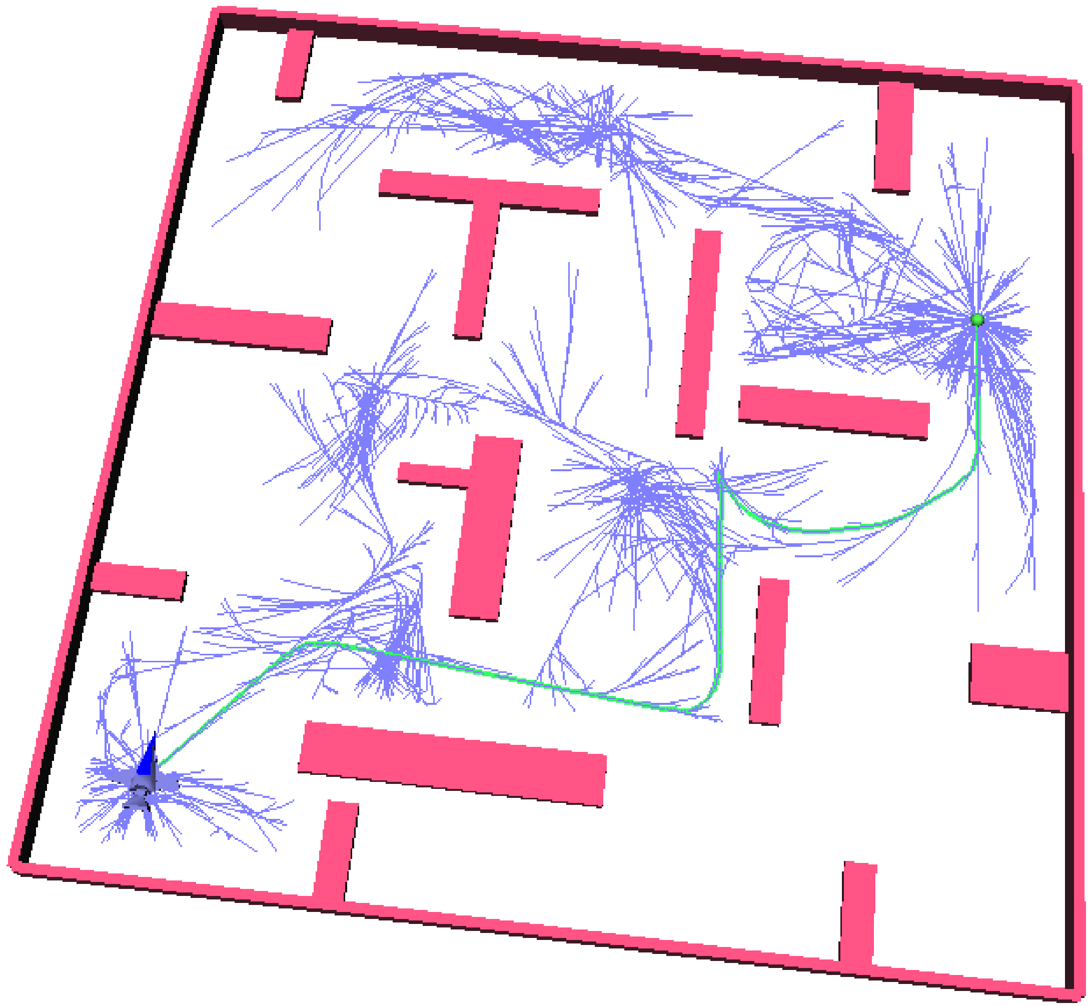
  assume arc cost ≥ 0
- goal test: predicate on nodes

Find:

- cheapest path to a goal node



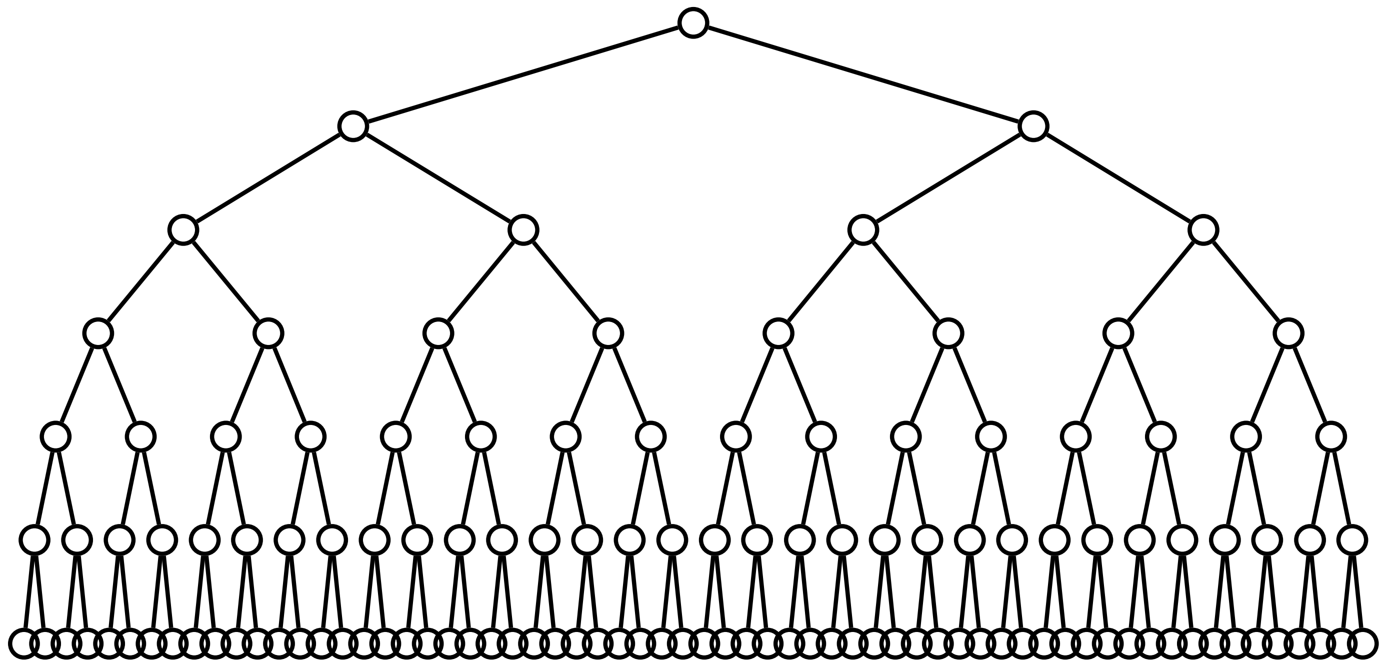
Example: Motion Planning



(S. LaValle)

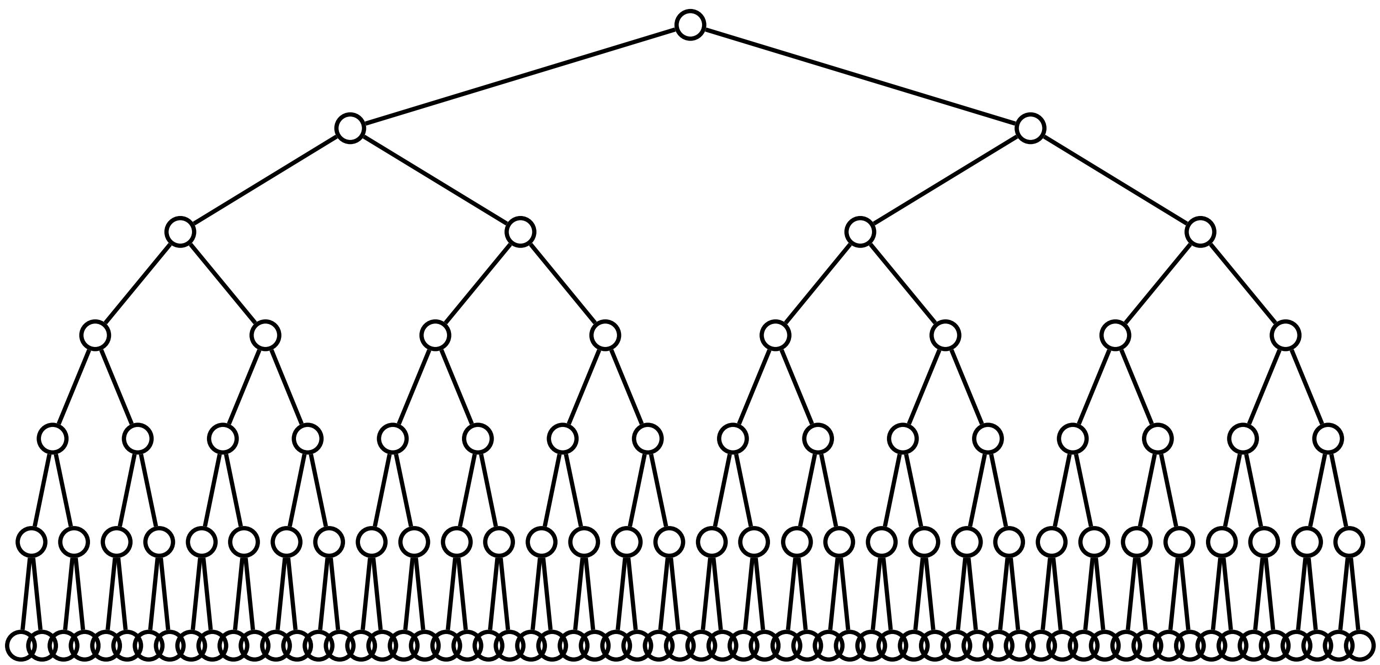
The Problem: Exponential Growth

Ever wonder why no one draws trees more than 3 levels deep?



The Problem: Exponential Growth

Ever wonder why no one draws trees more than 3 levels deep?



number of actions$^{\text{plan length}}$

$$14^{100} > 10^{80}$$

A Tale of Three Algorithms

1. Uniform Cost Search (Dijkstra, 1959)
2. A* Search (Hart, Nilsson, and Raphael, 1968)
3. Explicit Estimation Search (Thayer and Ruml, 2011)

Uniform Cost Search (Dijkstra, 1959)

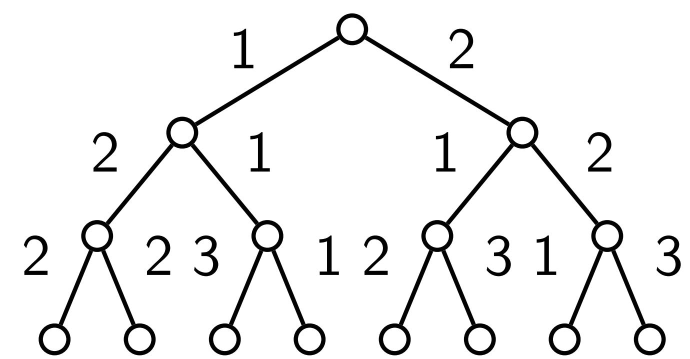
Uniform Cost Search (Dijkstra, 1959)

Explore nodes in increasing order of cost-so-far ($g(n)$):

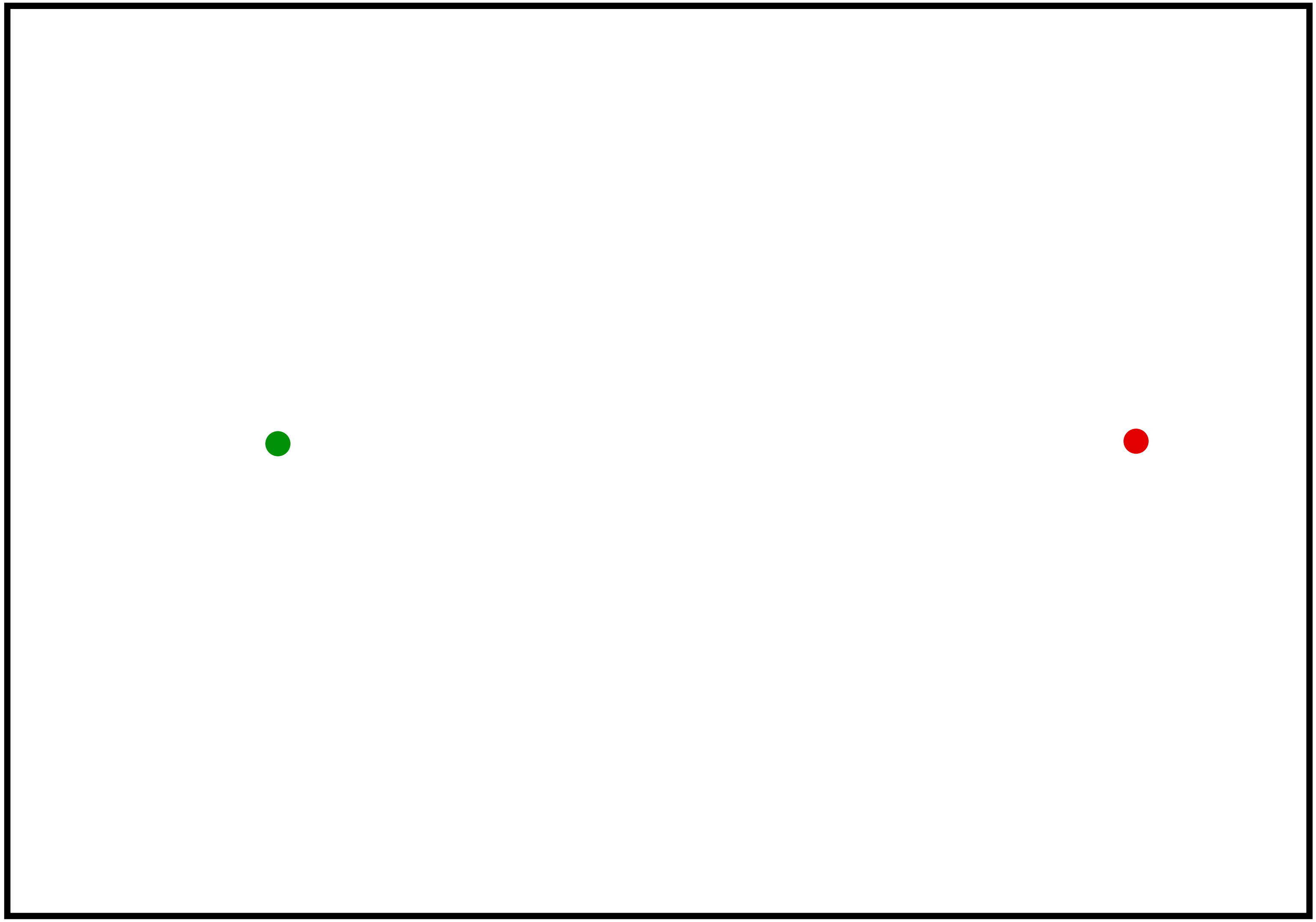
Uniform Cost Search (Dijkstra, 1959)

Explore nodes in increasing order of cost-so-far ($g(n)$):

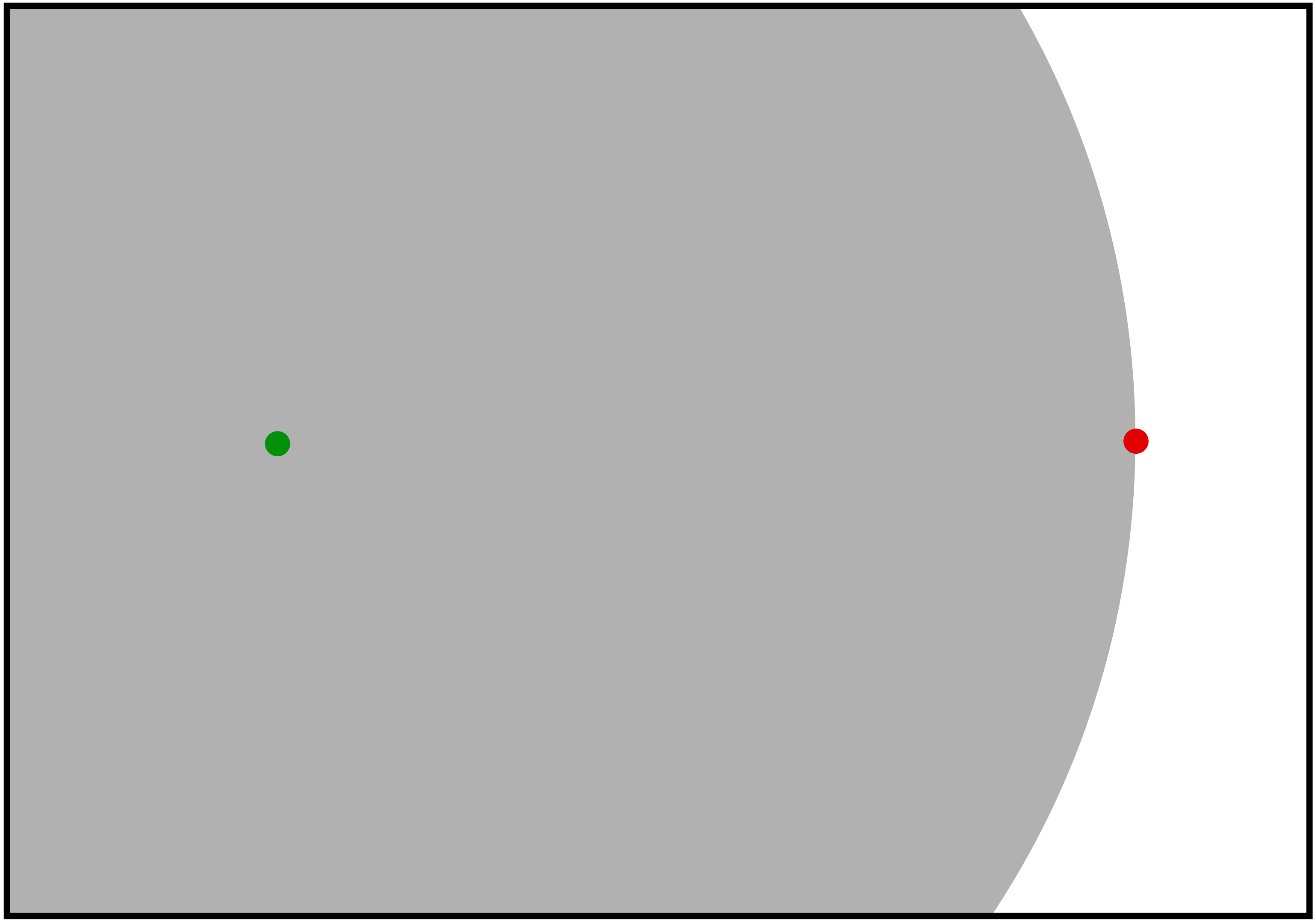
$Q \leftarrow$ ordered list containing the initial state
Loop
  If Q is empty, return failure
  Node $\leftarrow$ pop cheapest node off Q
  If *Node* is a goal, return it (or path to it)
  Children $\leftarrow$ Expand(*Node*).
  Merge *Children* into Q, keeping sorted by $g(n)$.



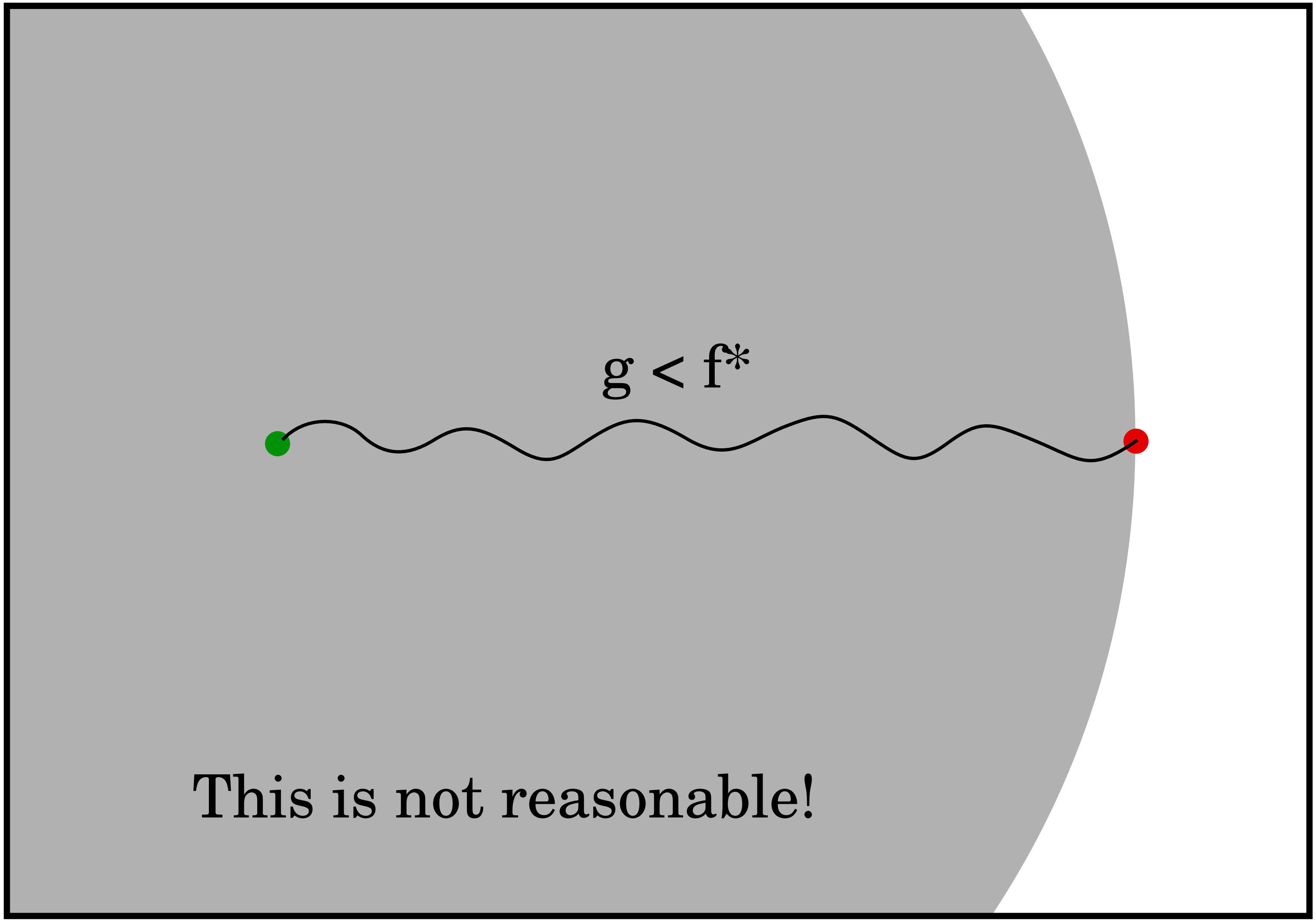
Dijkstra Behavior



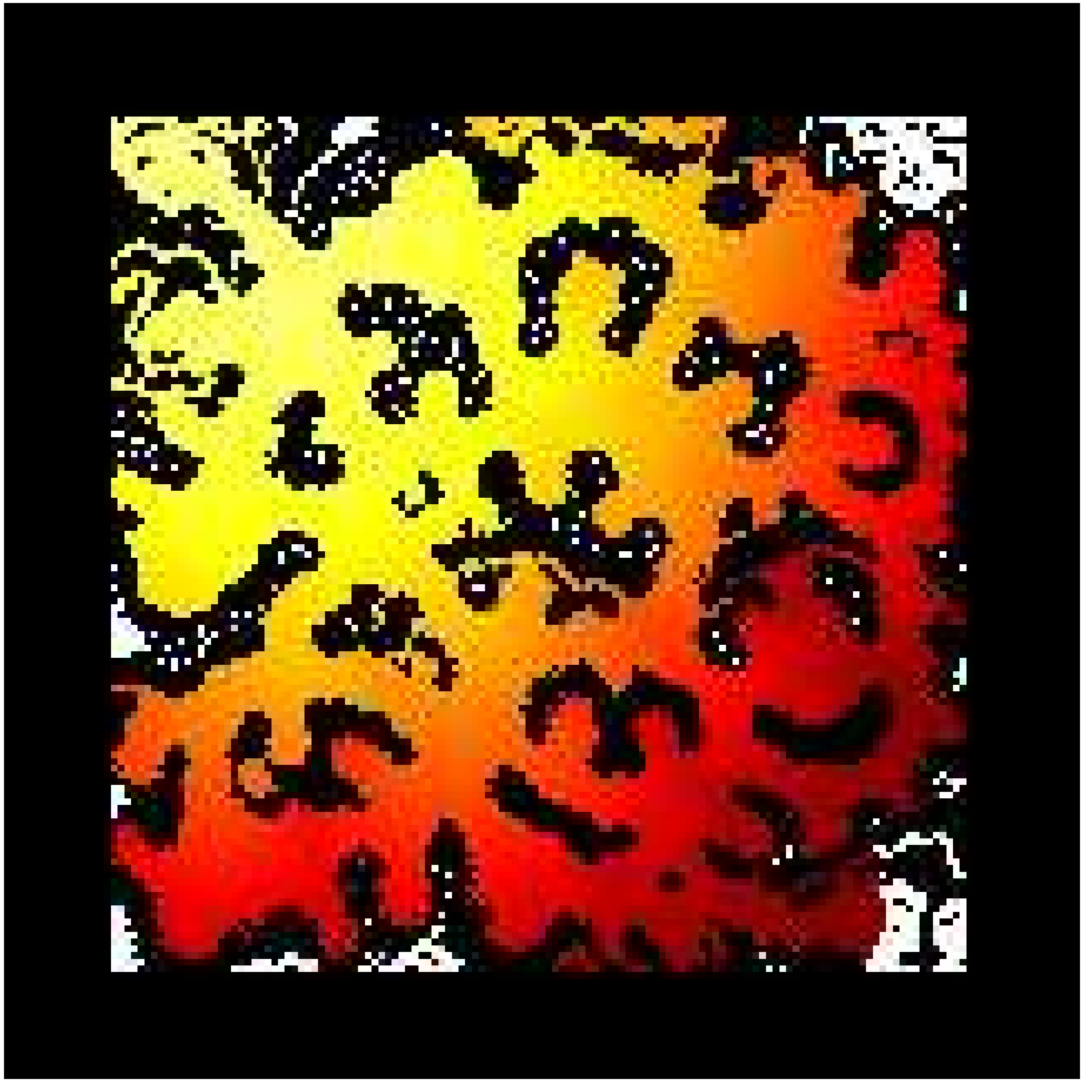
Dijkstra Behavior



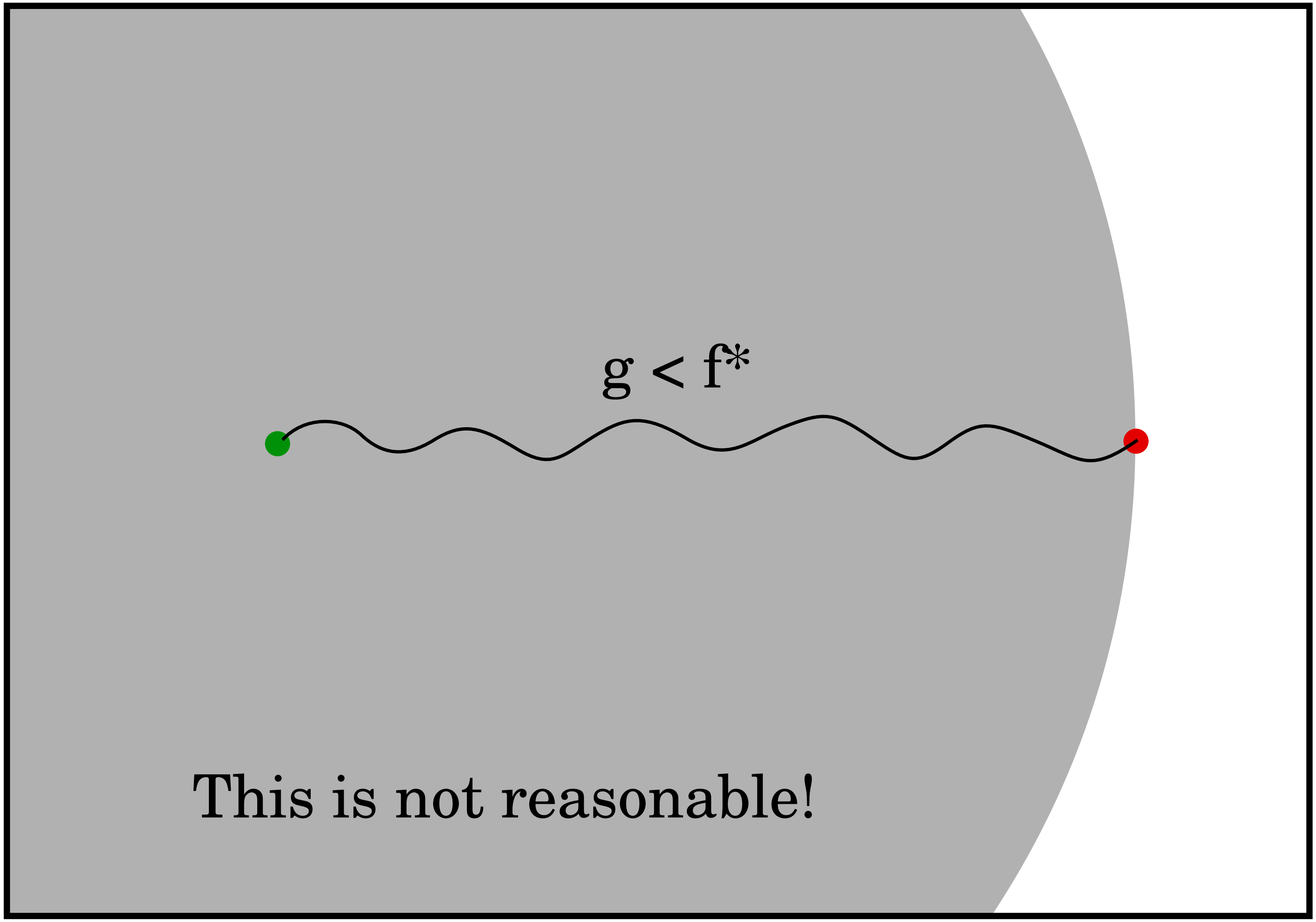
Dijkstra Behavior



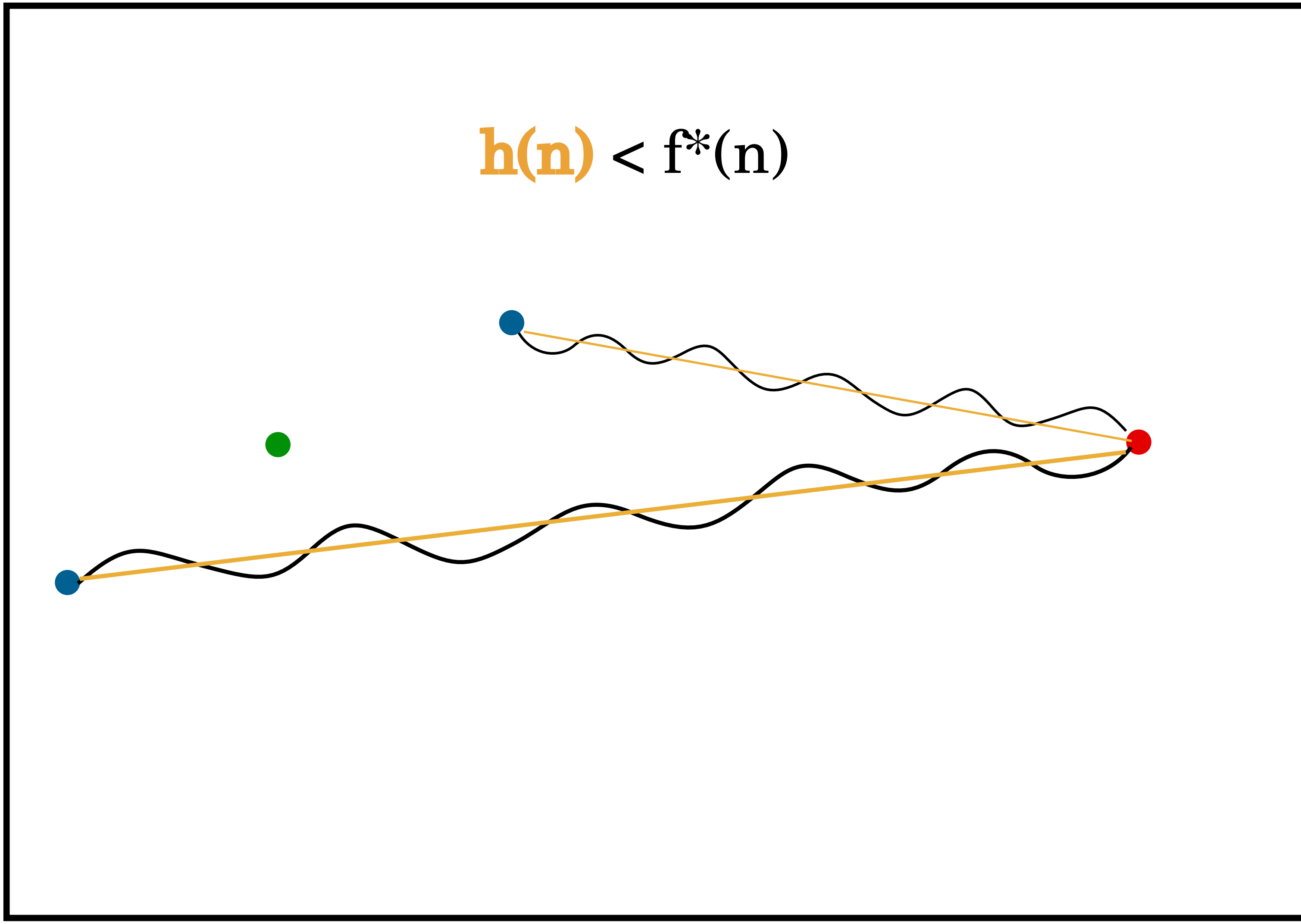
Dijkstra: Pathfinding in Warcraft



Problem-Specific Background Knowledge



Lower Bounds on Cost-to-go



Heuristic Search: A* (Hart, Nilsson, and Raphael, 1968)

Heuristic Search: A* (Hart, Nilsson, and Raphael, 1968)

Cost should include both cost-so-far and cost-to-go:

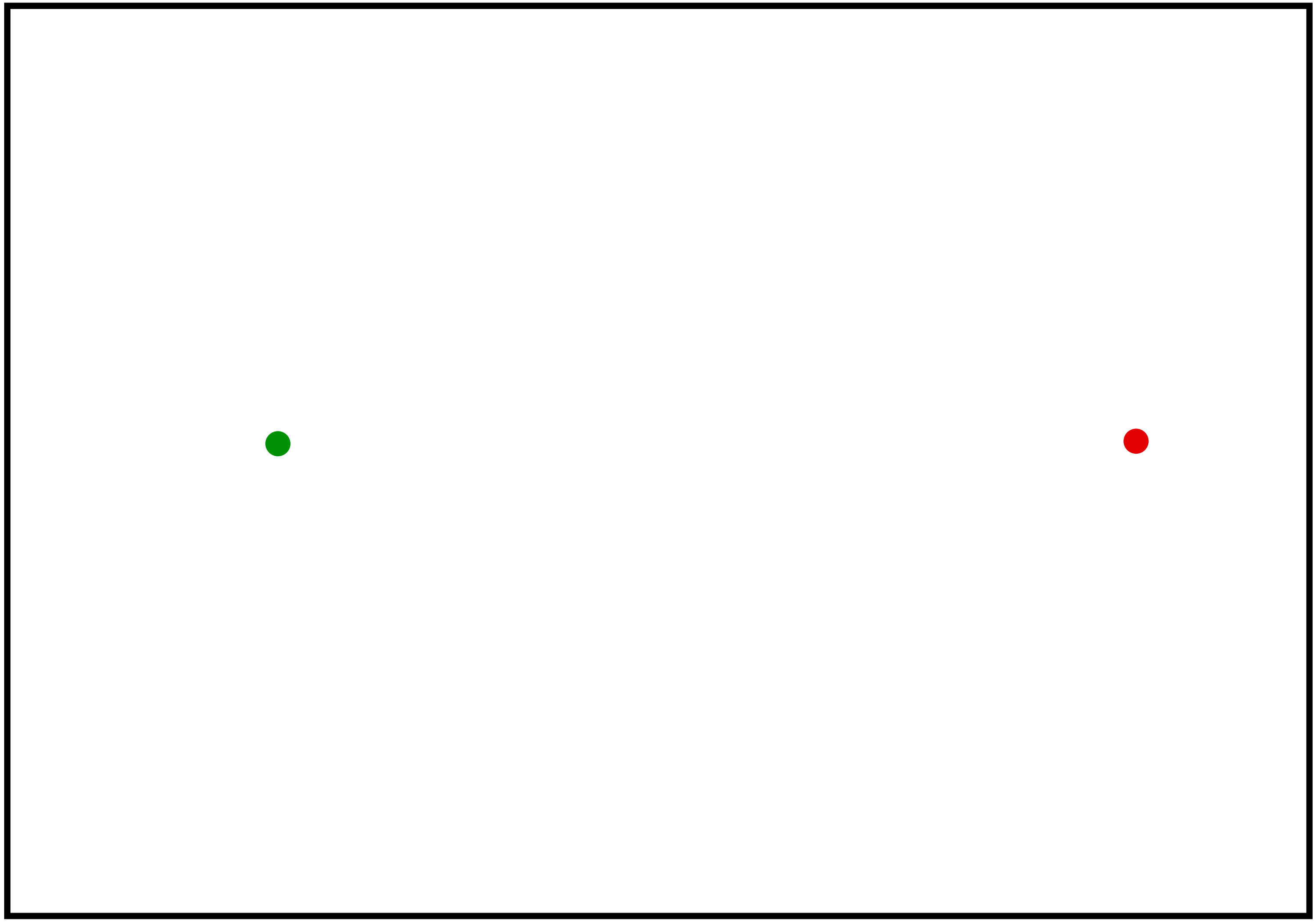
Heuristic Search: A* (Hart, Nilsson, and Raphael, 1968)

Cost should include both cost-so-far and cost-to-go:

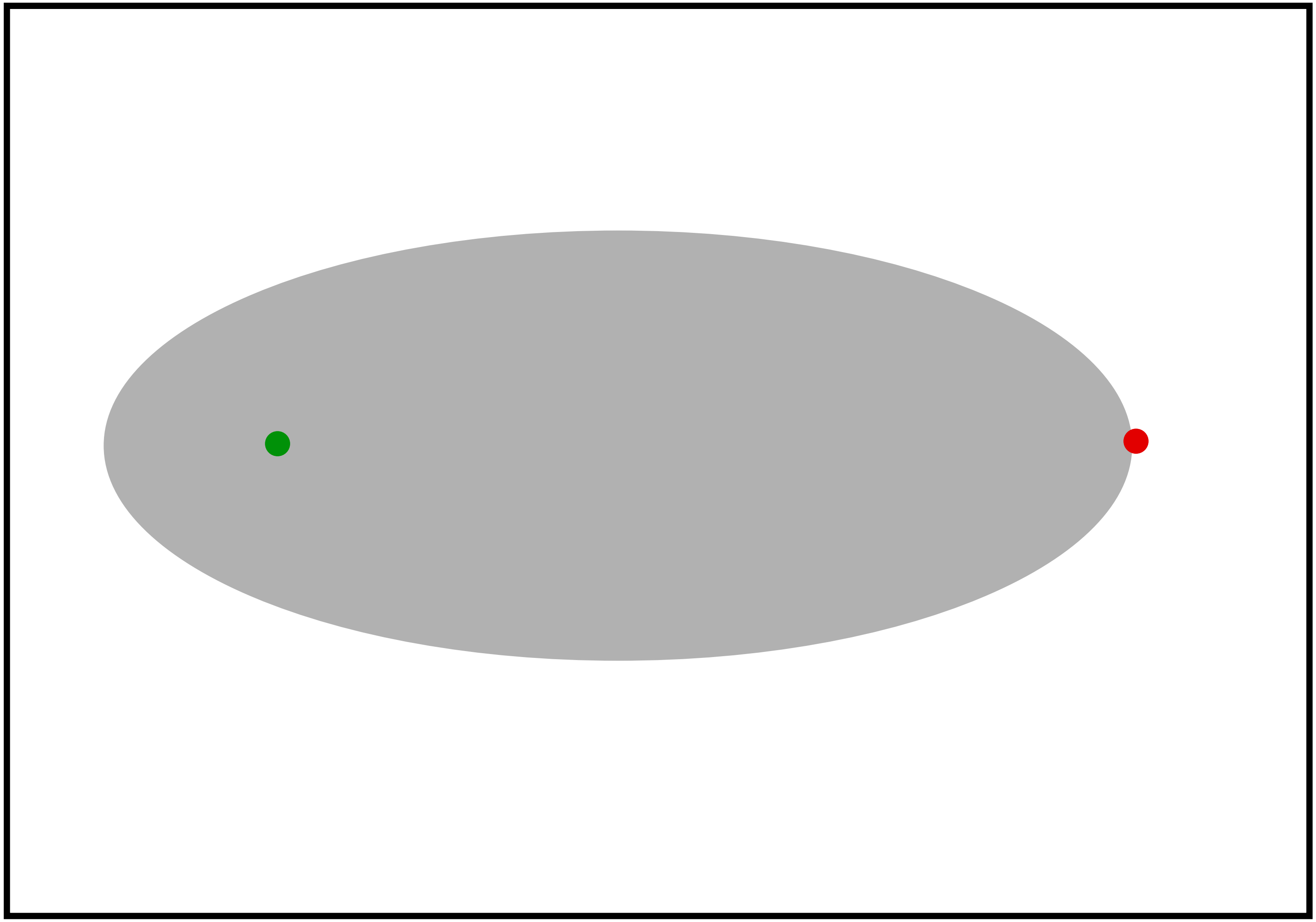
$Q \leftarrow$ ordered list containing the initial state
Loop
  If Q is empty, return failure
  Node $\leftarrow$ pop cheapest node off Q
  If *Node* is a goal, return it (or path to it)
  Children $\leftarrow$ Expand(*Node*)
  Merge *Children* into Q, keeping sorted by $f(n)$

$f(n) = g(n) + h(n)$
$g(n) =$ cost incurred so far
$h(n) =$ lower bound on cost to goal

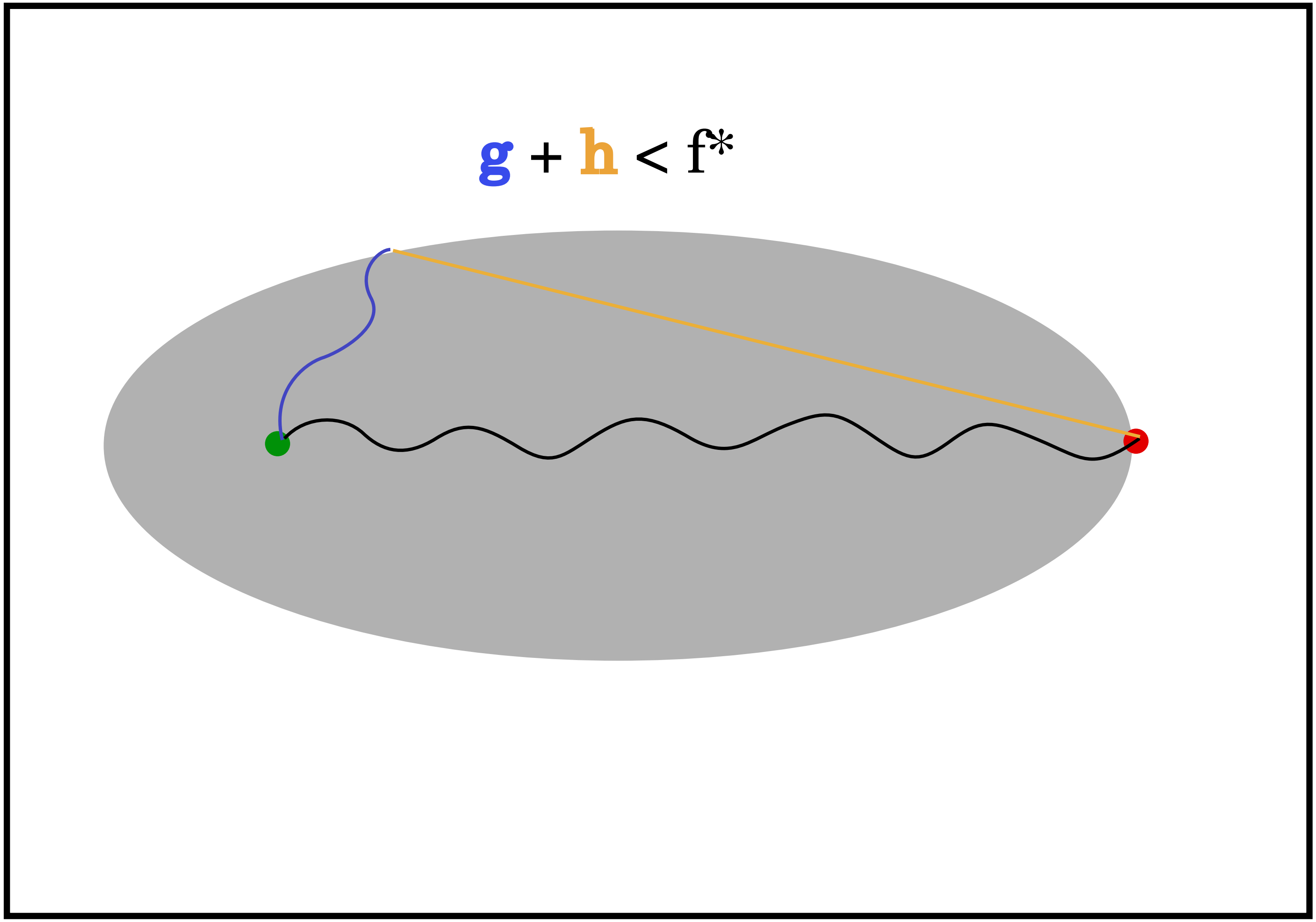
A* Behavior



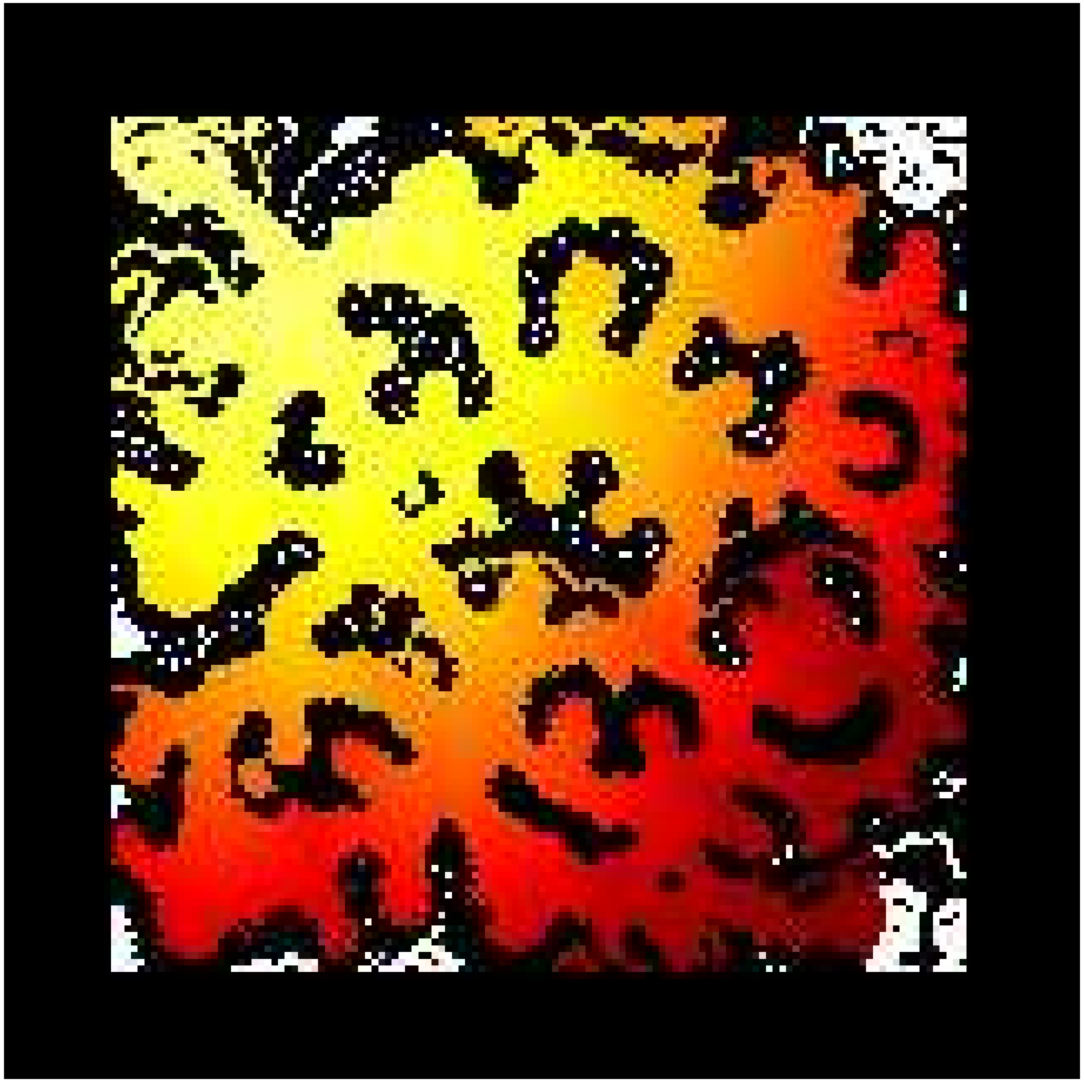
A* Behavior



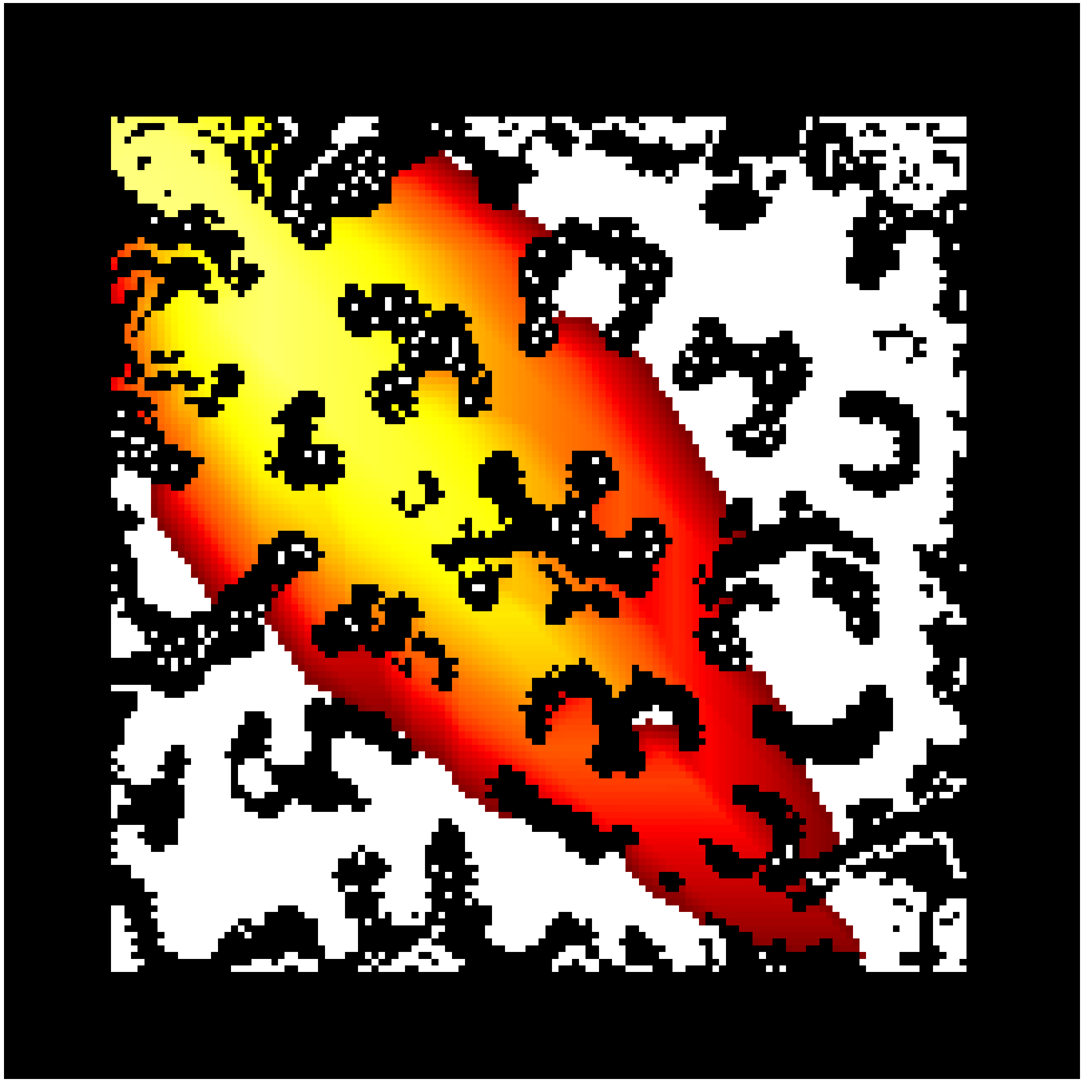
A* Behavior



Dijkstra vs A*: Pathfinding in Warcraft



Dijkstra vs A*: Pathfinding in Warcraft



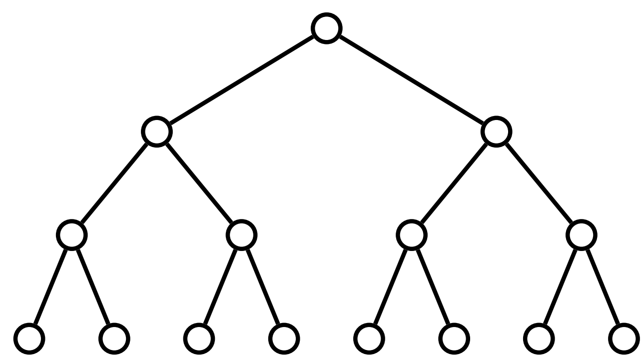
Quick Note: A Lower Bound on Solution Cost

$$f(n) = g(n) + h(n)$$

$g(n)$ is actual cost-so-far, so
when $h(n)$ is a lower bound on cost-to-go,
$f(n)$ is a lower bound on final solution cost

lowest $f(n)$ on frontier gives lower bound for entire problem!

$$best_f = \operatorname*{argmin}_{n \in open} f(n)$$



Problems with A*

Problems with A*

A* takes exponential memory

Problems with A*

A* takes exponential memory

Can sometimes be fixed: see 'iterative deepening'

Problems with A*

A* takes exponential memory

Can sometimes be fixed: see 'iterative deepening'

A* takes exponential time

Problems with A*

A* takes exponential memory

Can sometimes be fixed: see 'iterative deepening'

A* takes exponential time

We must trade cost for time.

A New Generation of Problem Settings

optimal: minimize solution cost

must expand all with $f(n) < f^*(opt)$

A New Generation of Problem Settings

optimal: minimize solution cost
must expand all with $f(n) < f^*(opt)$

greedy: minimize solving time

bounded suboptimal: minimize time subject to relative cost bound (factor of optimal)

bounded cost: minimize time subject to absolute cost bound

contract: minimize cost subject to absolute time bound

utility function: maximize utility function of cost and time
eg, goal achievement time =
plan makespan + search time

A New Generation of Problem Settings

optimal: minimize solution cost

must expand all with $f(n) < f^*(opt)$

greedy: minimize solving time

bounded suboptimal: minimize time subject to relative cost bound (factor of optimal)

bounded cost: minimize time subject to absolute cost bound

contract: minimize cost subject to absolute time bound

utility function: maximize utility function of cost and time

eg, goal achievement time =

plan makespan + search time

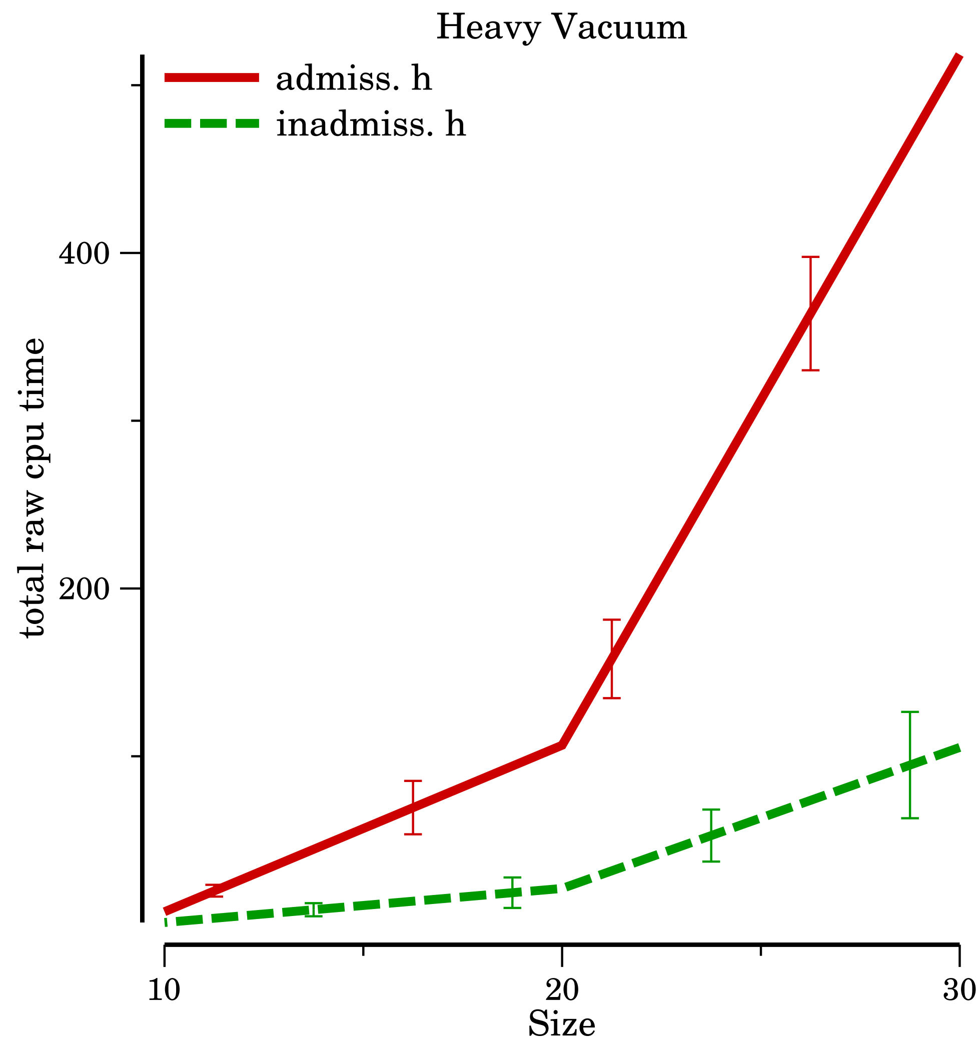
Bounded Suboptimal Search

The Intuition Behind EES (Thayer and Ruml, 2011)

- unbiased estimates can be more informed than lower bounds
- nearest goal is the easiest to find

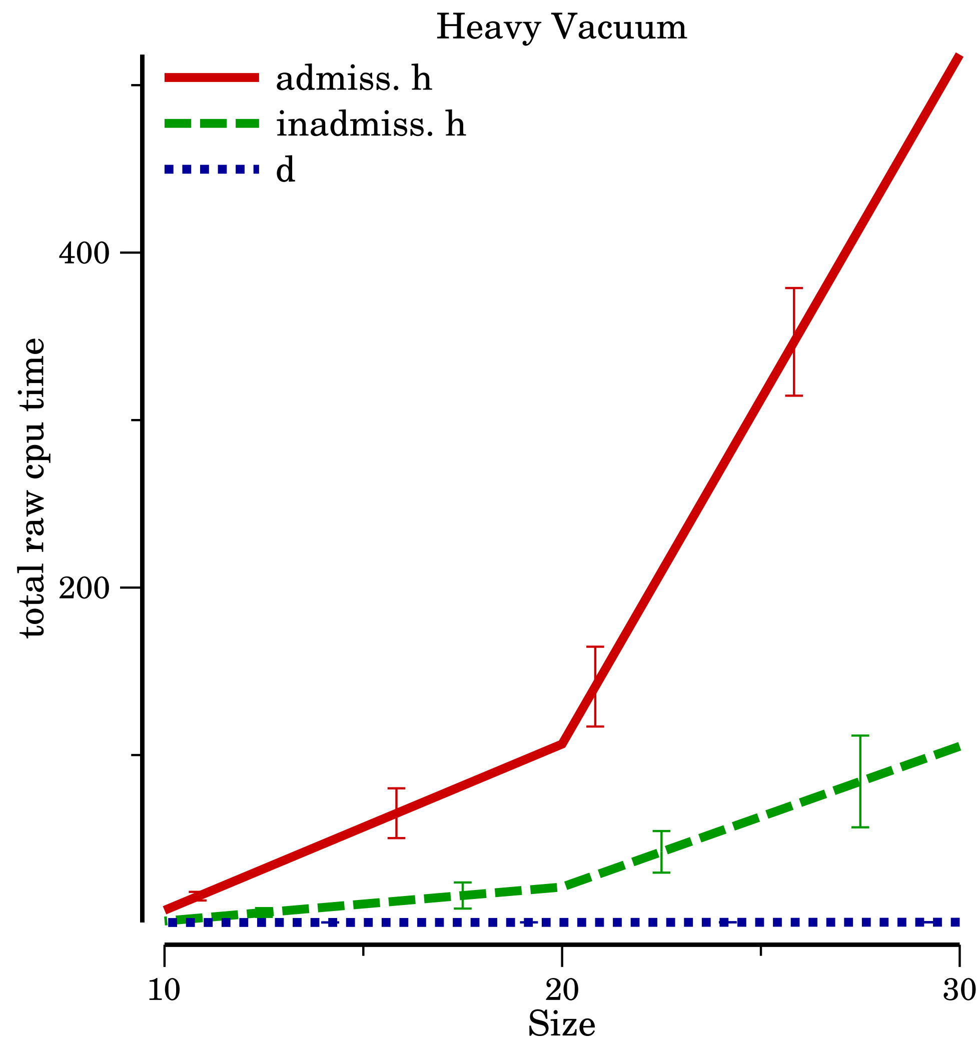
The Intuition Behind EES (Thayer and Ruml, 2011)

- unbiased estimates can be more informed than lower bounds
- nearest goal is the easiest to find



The Intuition Behind EES (Thayer and Ruml, 2011)

- unbiased estimates can be more informed than lower bounds
- nearest goal is the easiest to find



The Intuition Behind EES (Thayer and Ruml, 2011)

- unbiased estimates can be more informed than lower bounds
- nearest goal is the easiest to find

Explicit Estimation Search (EES) combines these ideas

to minimize solving time subject to cost $\leq w \cdot$optimal:

pursue the nearest goal within the bound

need more information than just lower bound on cost $(h(n))$!

Three Heuristic Sources of Information

1. h: a lower bound on cost-to-go
   $f(n) = g(n) + h(n)$
   the traditional optimal A* lower bound

Three Heuristic Sources of Information

1. h: a lower bound on cost-to-go

   $f(n) = g(n) + h(n)$

   the traditional optimal A* lower bound

2. $\widehat{h}$: an estimate of cost-to-go

   unbiased estimates can be more informed

   $\widehat{f}(n) = g(n) + \widehat{h}(n)$

   (Thayer and Ruml, ICAPS-11)

Three Heuristic Sources of Information

1. h: a lower bound on cost-to-go

   $f(n) = g(n) + h(n)$

   the traditional optimal A* lower bound

2. $\widehat{h}$: an estimate of cost-to-go

   unbiased estimates can be more informed

   $\widehat{f}(n) = g(n) + \widehat{h}(n)$

   (Thayer and Ruml, ICAPS-11)

3. $\widehat{d}$: an estimate of distance-to-go

   nearest goal is the easiest to find

   (Pearl and Kim, IEEE PAMI 1982,
   Thayer et al, ICAPS-09)

Three Heuristic Sources of Information

1. h: a lower bound on cost-to-go

   $f(n) = g(n) + h(n)$

   the traditional optimal A* lower bound

2. $\widehat{h}$: an estimate of cost-to-go

   unbiased estimates can be more informed

   $\widehat{f}(n) = g(n) + \widehat{h}(n)$

   (Thayer and Ruml, ICAPS-11)

3. $\widehat{d}$: an estimate of distance-to-go

   nearest goal is the easiest to find

   (Pearl and Kim, IEEE PAMI 1982,
   Thayer et al, ICAPS-09)

pursue the nearest goal within the bound

$best_f$: open node giving lower bound on cost

$$\underset{n \in open}{\operatorname{argmin}} f(n)$$

Finding $best_{\widehat{d}}$

$best_f$: open node giving lower bound on cost

$$\operatorname*{argmin}_{n \in open} f(n)$$

$best_{\widehat{f}}$: open node giving estimated optimal cost

$$\operatorname*{argmin}_{n \in open} \widehat{f}(n)$$

$best_f$: open node giving lower bound on cost

$$\underset{n \in open}{\operatorname{argmin}} f(n)$$

$best_{\widehat{f}}$: open node giving estimated optimal cost

$$\underset{n \in open}{\operatorname{argmin}} \widehat{f}(n)$$

pursue the nearest goal within the bound

$best_{\widehat{d}}$: estimated w-suboptimal node with minimum $\widehat{d}$

$$\underset{n \in open \wedge \widehat{f}(n) \leq w \cdot \widehat{f}(best_{\widehat{f}})}{\operatorname{argmin}} \widehat{d}(n)$$

EES Expansion Order

$best_f$: open node giving lower bound on cost
$best_{\widehat{f}}$: open node giving estimated optimal cost
$best_{\widehat{d}}$: estimated w-suboptimal node with minimum $\widehat{d}$

node to expand next:

1. pursue the shortest solution that is within the bound
2.
3.

in other words:

1. $best_{\widehat{d}}$
2.
3.

EES Expansion Order

$best_f$: open node giving lower bound on cost
$best_{\widehat{f}}$: open node giving estimated optimal cost
$best_{\widehat{d}}$: estimated w-suboptimal node with minimum $\widehat{d}$

node to expand next:

1. pursue the shortest solution that is within the bound
2.
3.

in other words:

1. **if** $\widehat{f}(best_{\widehat{d}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{d}}$
2.
3.

EES Expansion Order

$best_f$: open node giving lower bound on cost
$best_{\widehat{f}}$: open node giving estimated optimal cost
$best_{\widehat{d}}$: estimated w-suboptimal node with minimum $\widehat{d}$

node to expand next:

1. pursue the shortest solution that is within the bound
2. pursue the estimated optimal solution
3.

in other words:

1. **if** $\widehat{f}(best_{\widehat{d}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{d}}$
2. **else if** $\widehat{f}(best_{\widehat{f}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{f}}$
3.

EES Expansion Order

$best_f$: open node giving lower bound on cost
$best_{\widehat{f}}$: open node giving estimated optimal cost
$best_{\widehat{d}}$: estimated w-suboptimal node with minimum $\widehat{d}$

node to expand next:

1. pursue the shortest solution that is within the bound
2. pursue the estimated optimal solution
3. raise the lower bound on optimal solution cost

in other words:

1. **if** $\widehat{f}(best_{\widehat{d}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{d}}$
2. **else if** $\widehat{f}(best_{\widehat{f}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{f}}$
3. **else** $best_f$

see paper for further justification

EES Respects the Suboptimality Bound

how does $\widehat{f}(n) \leq w \cdot f(best_f)$ ensure the suboptimality bound?

EES Respects the Suboptimality Bound

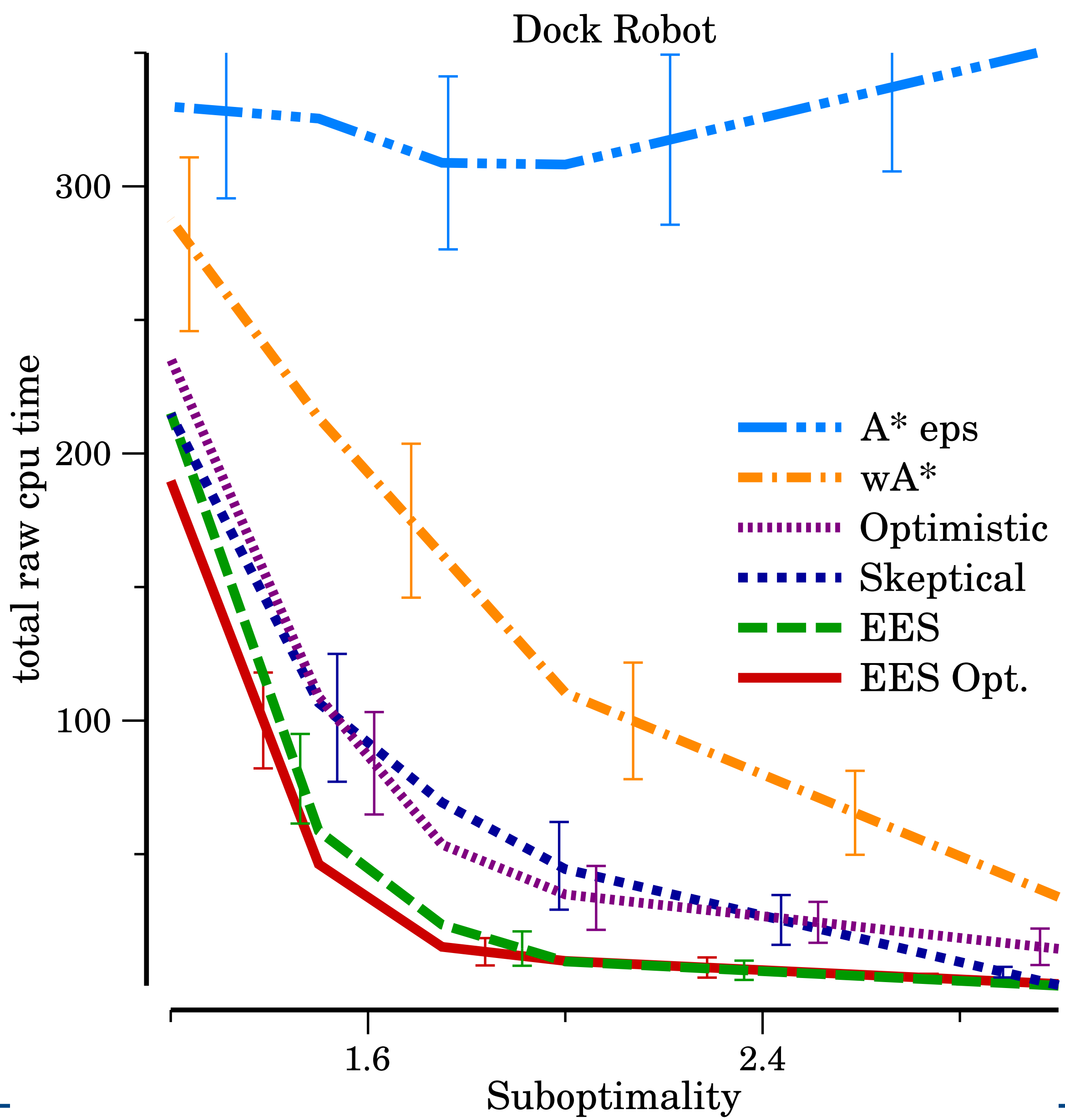
how does $\widehat{f}(n) \leq w \cdot f(best_f)$ ensure the suboptimality bound?

$$
\begin{array}{rcll}
f(n) & \leq & \widehat{f}(n) & f(n) \text{ is a lower bound for } n \\
\widehat{f}(n) & \leq & w \cdot f(best_f) & \text{expansion criterion} \\
w \cdot f(best_f) & \leq & w \cdot f^*(opt) & \text{because } f(best_f) \text{ is a lower} \\
 & & & \text{bound for the entire problem} \\
\hline
f(n) & \leq & w \cdot f^*(opt) & \text{suboptimality bound}
\end{array}
$$

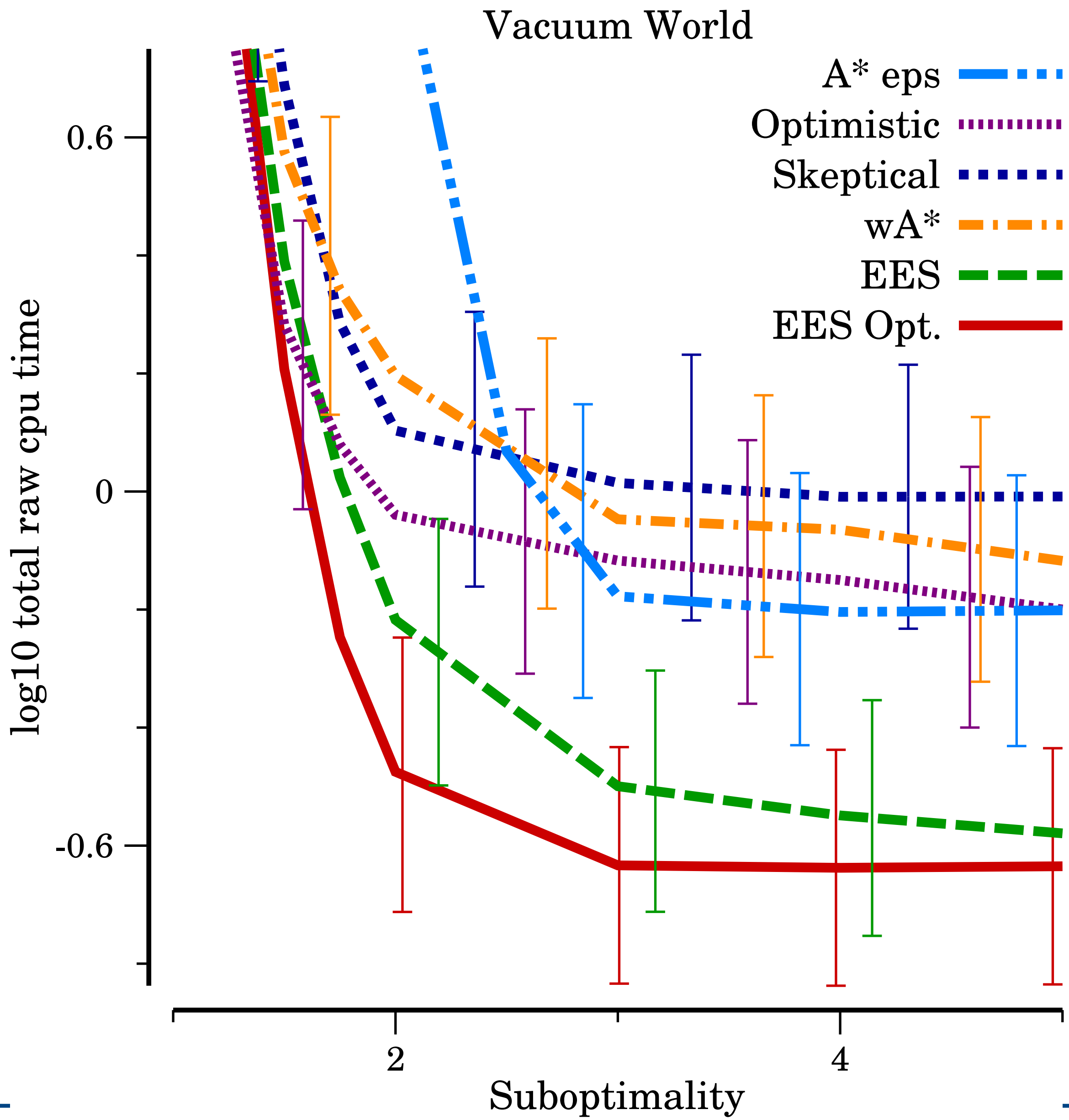
EES Performance

bounded suboptimal search:
minimize time subject to
relative cost bound (factor of optimal)

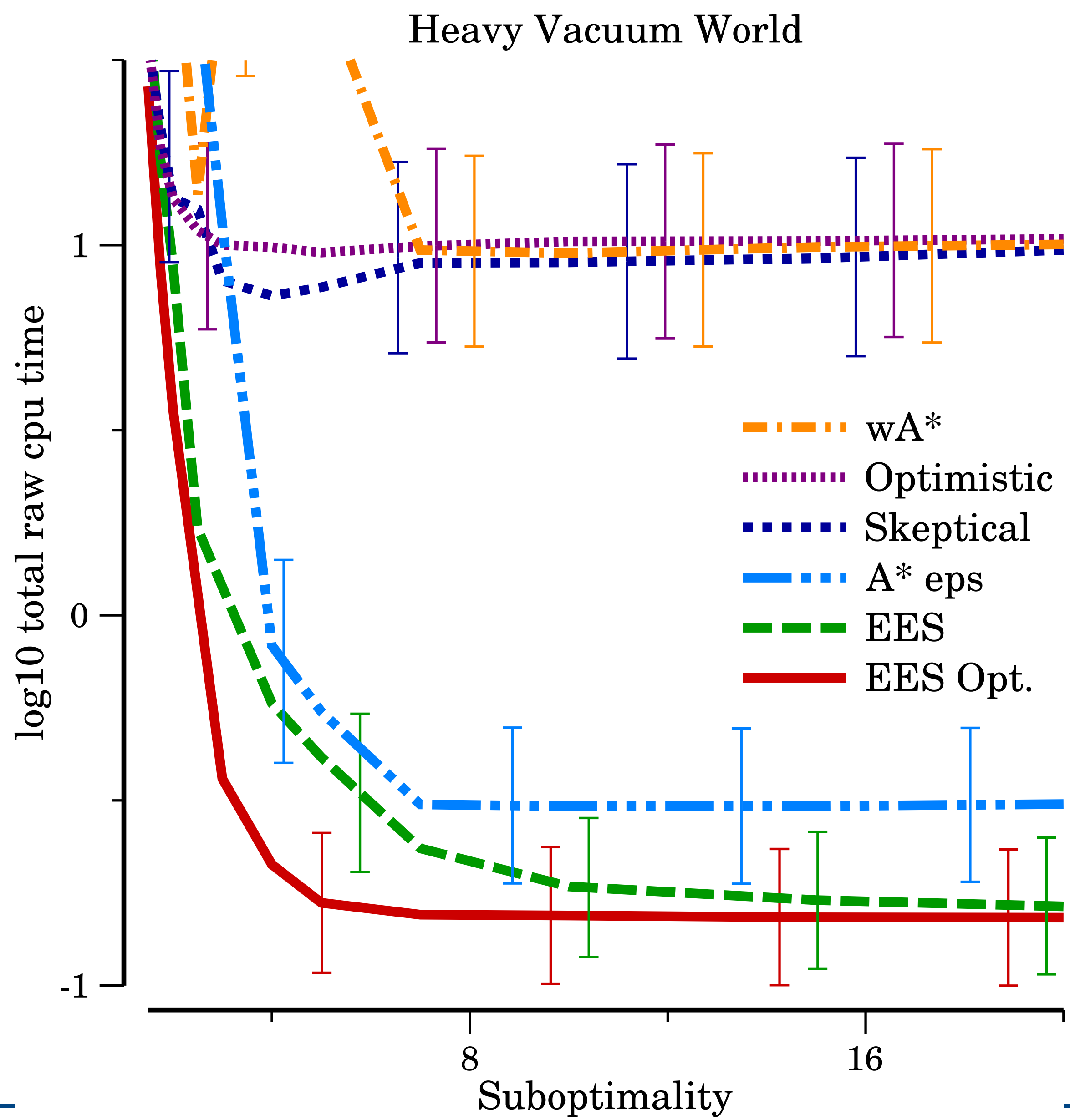
EES Performance



EES Performance



EES Performance



Conclusion

Summary

Who's afraid of big bad NP-hardness?

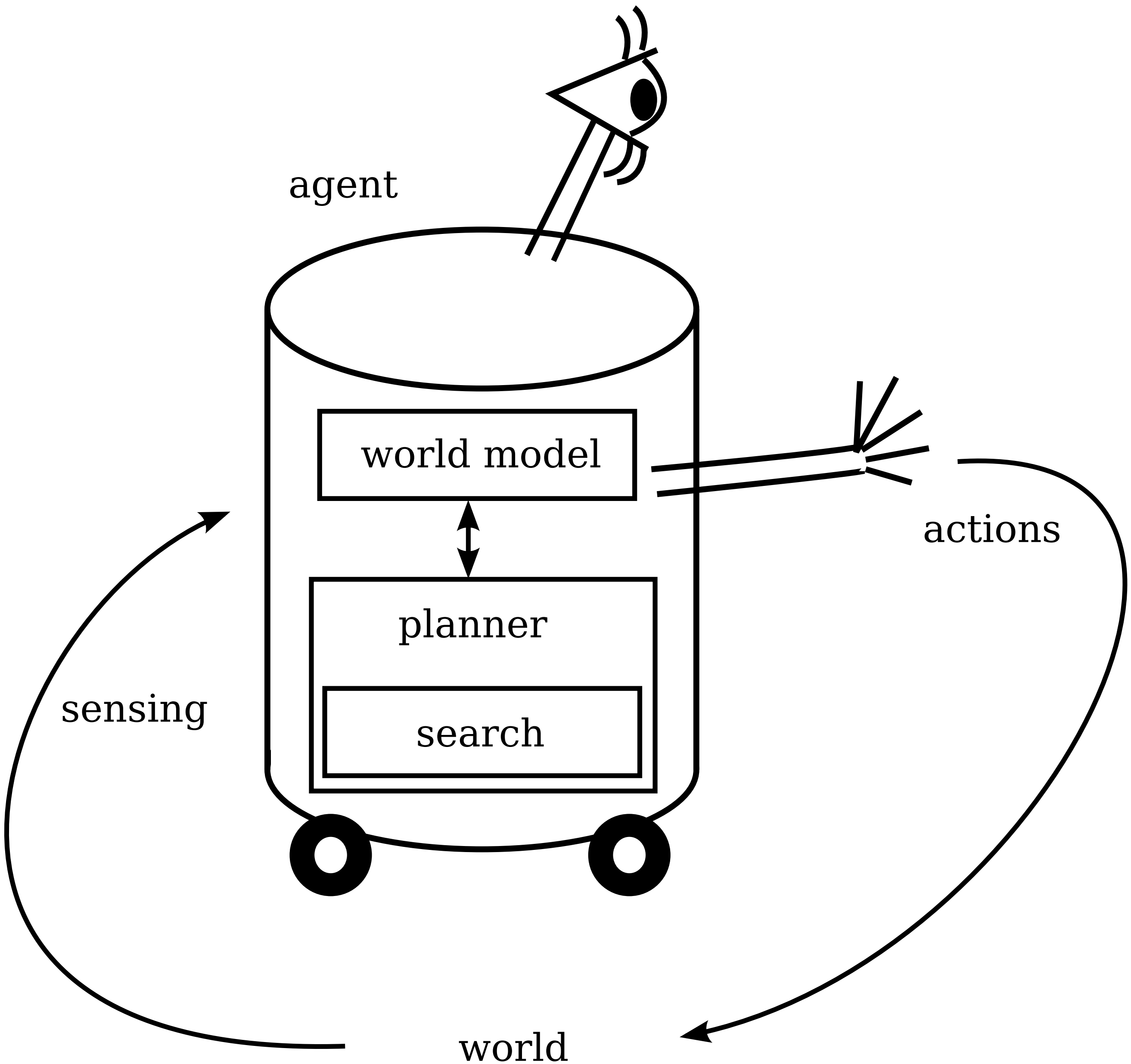
bounded suboptimal planning

- fast solving
- control over cost

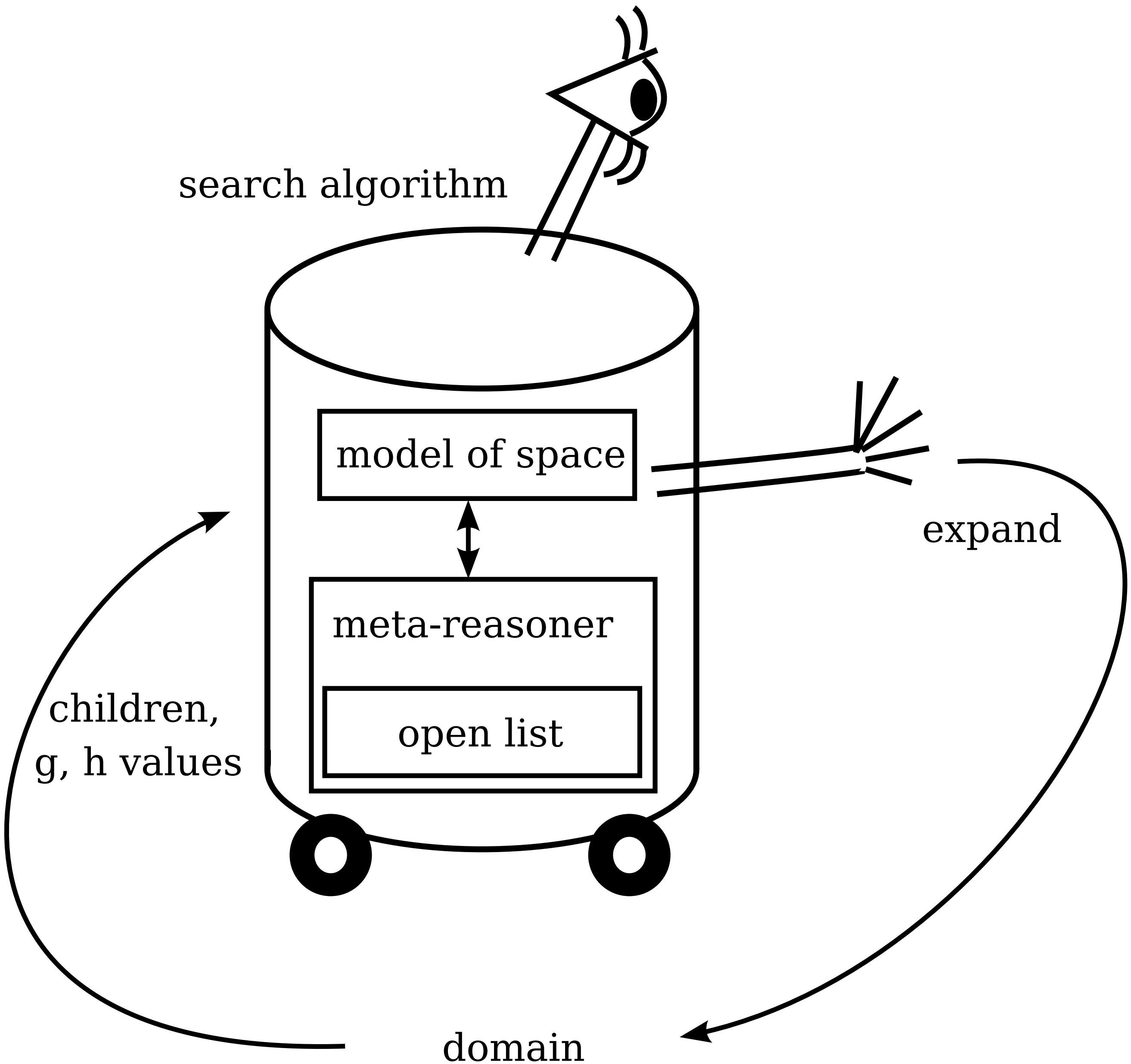
a new generation of planning algorithms

- beyond optimal
- estimates, in addition to lower bounds
- new sources of heuristic information
- principled, provable performance

The AI Vision

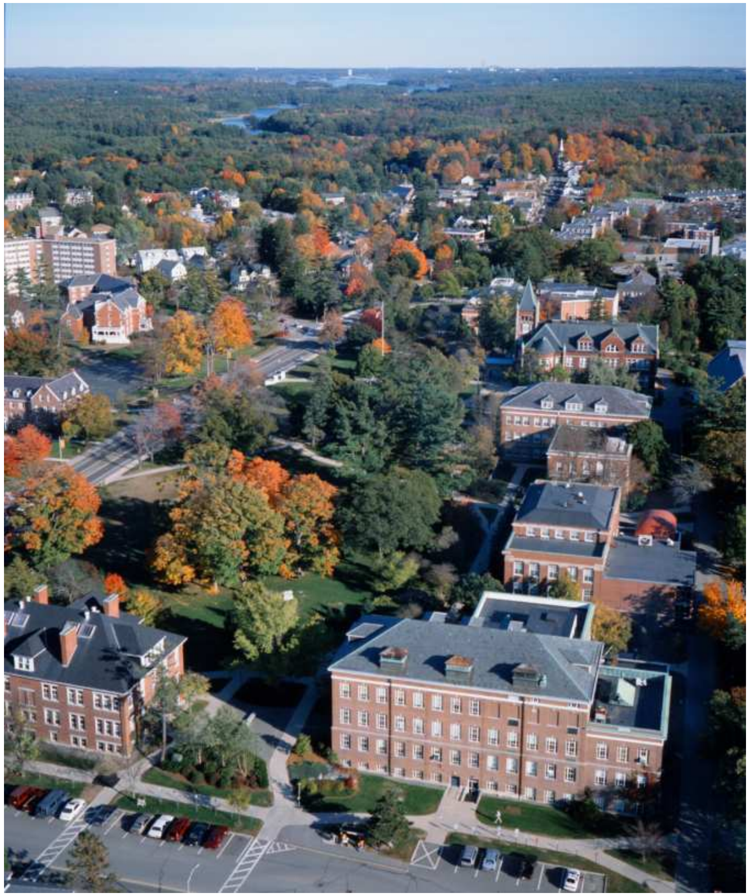


Search Algorithms as Agents



The University of New Hampshire

tell your students to apply to grad school in CS at UNH!



- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking, bioinformatics