# Planning Algorithms: When Optimal Is Just Not Good Enough

Wheeler Ruml

University of New Hampshire

Department of Computer Science

Joint work with the UNH AI Group. Support from NSF and DARPA.

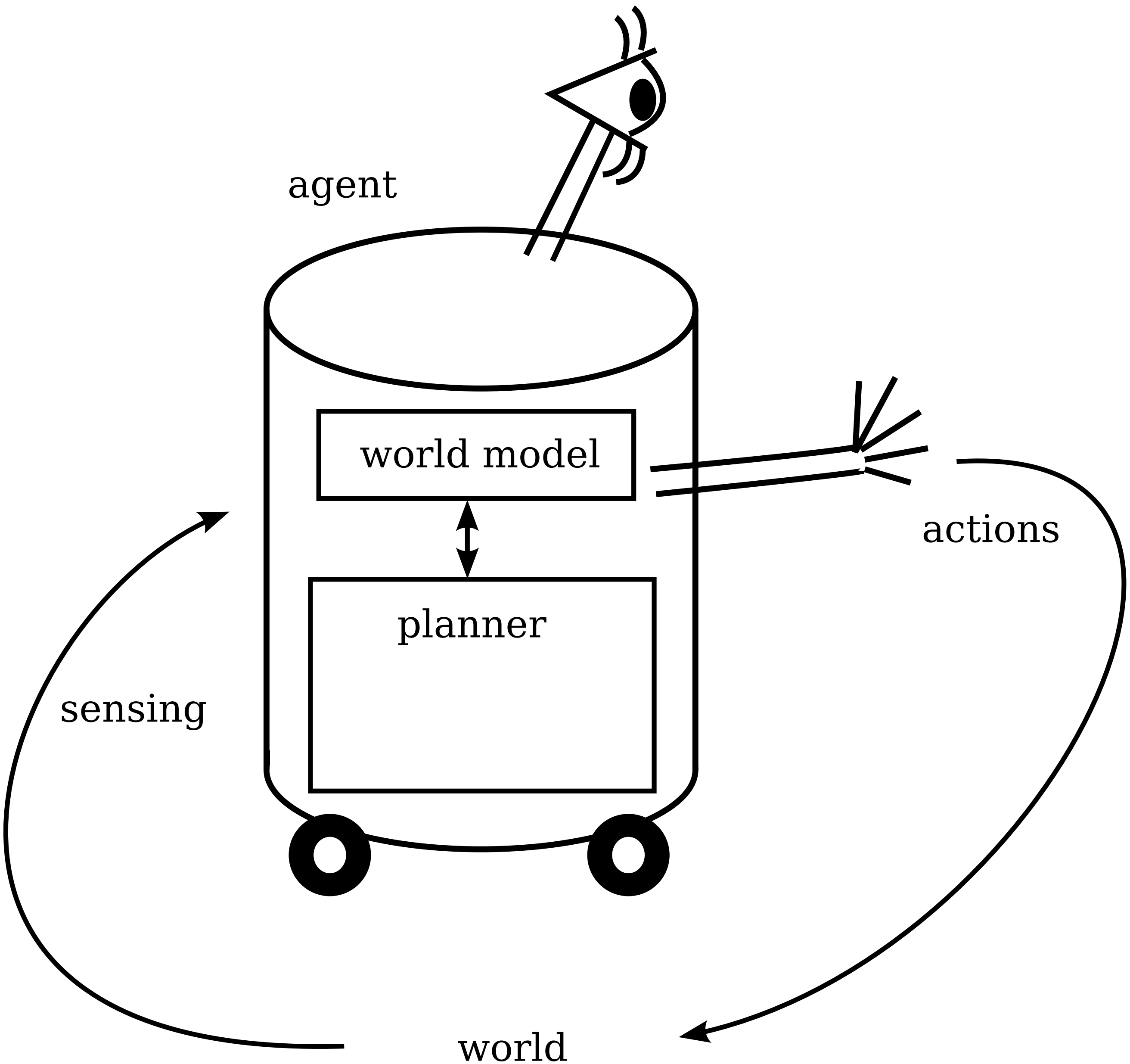
# Robot Time!





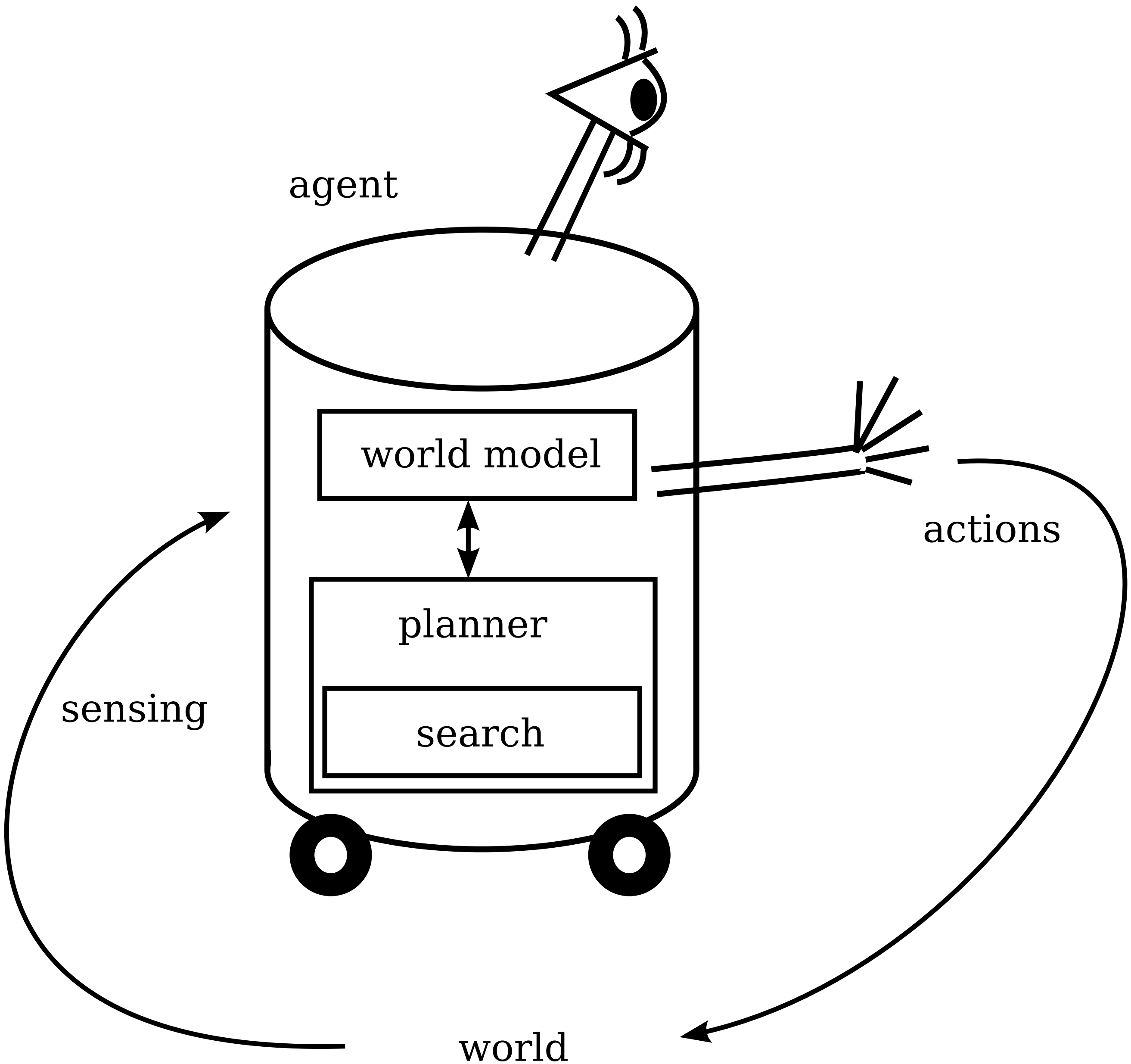
# The Role of Planning





# The Role of Planning





# The Roles of Planning



- autonomy
  - exploration
  - transportation
  - manufacturing
  - autonomic systems


- decision support
  - operations management
  - personal health
  - eldercare
  - education

# 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:

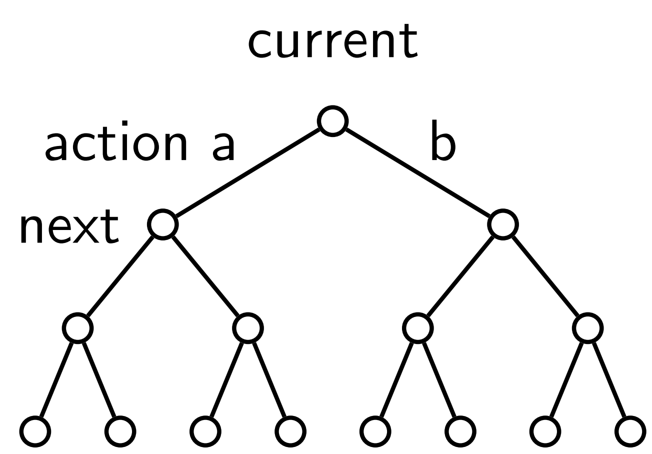
- cheapest plan

Given:

- start state: an explicit node
- expand function:
  lazily generate children and their costs
- goal test: predicate on nodes

Find:

- cheapest path to a goal node

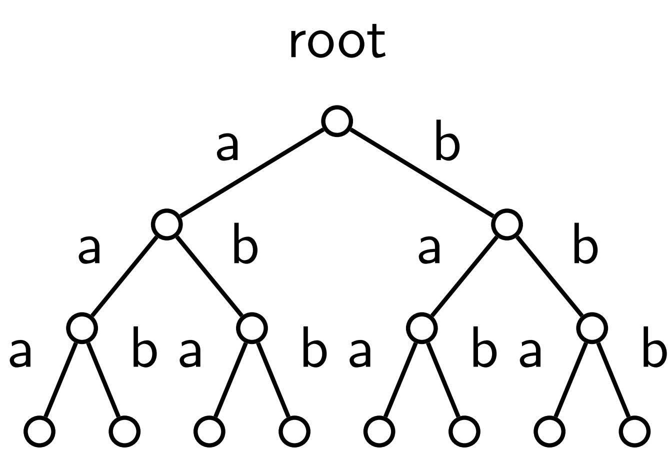


Given:

- start state: an explicit node
- expand function:
  lazily generate children and their costs
- goal test: predicate on nodes

Find:

- cheapest path to a goal node

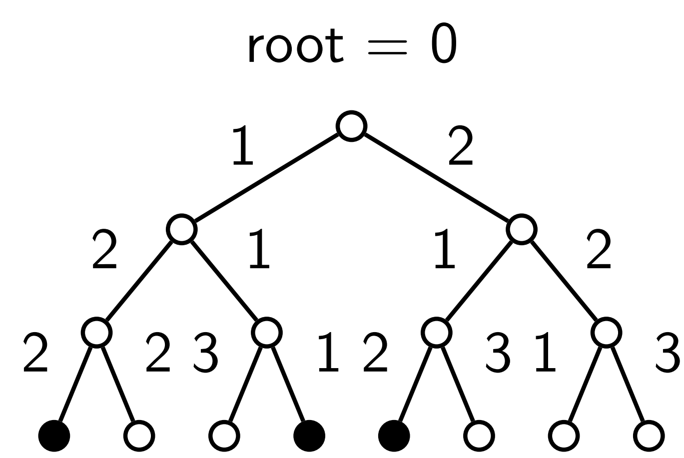


Given:

- start state: an explicit node
- expand function:
  lazily generate children and their costs
- goal test: predicate on nodes

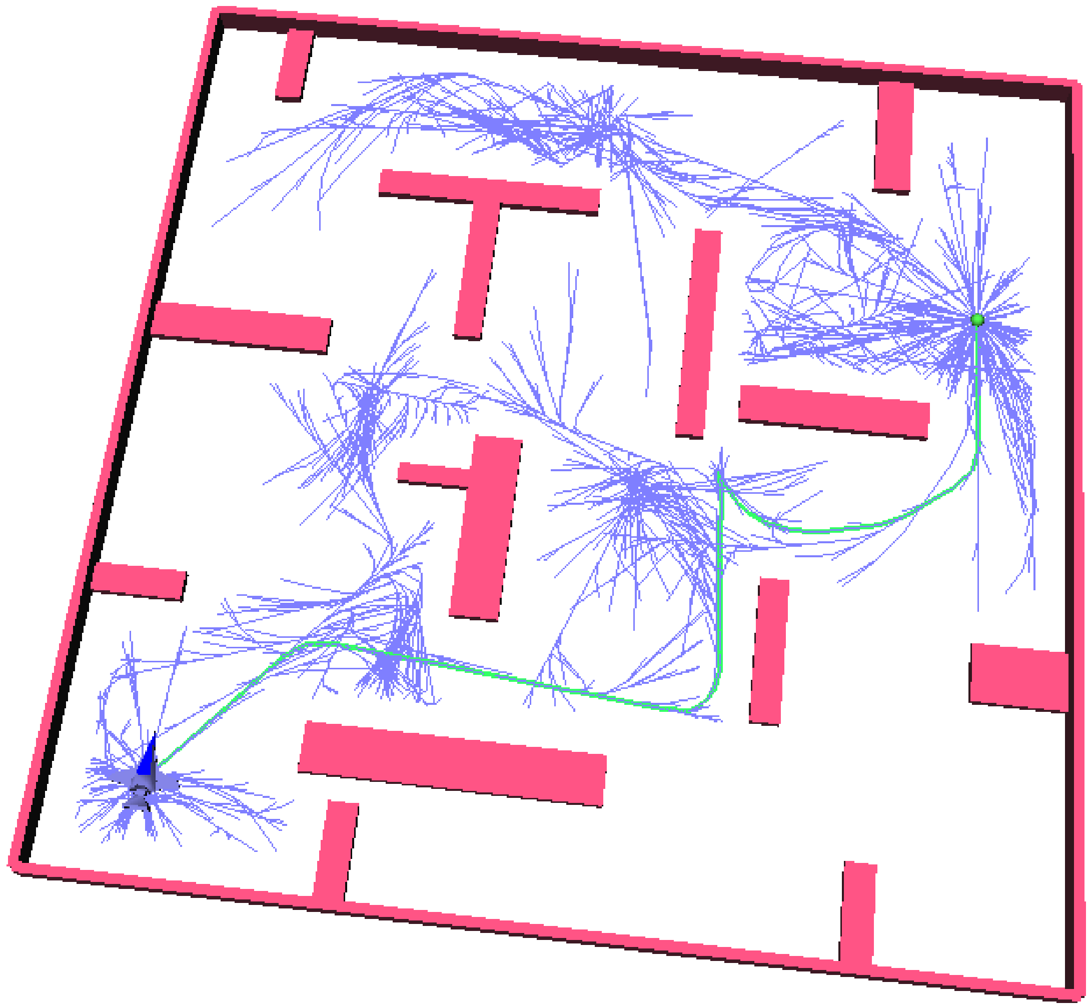
Find:

- cheapest path to a goal node



# Example: Motion Planning





(S. LaValle)

1. Uniform Cost Search (Dijkstra, 1959)

2. A* Search (Hart, Nilsson, and Raphael, 1968)

3. Explicit Estimation Search (Thayer and Ruml, 2011)

# Optimal Planning

# Uniform Cost Search (Dijkstra, 1959)

# Uniform Cost Search (Dijkstra, 1959)



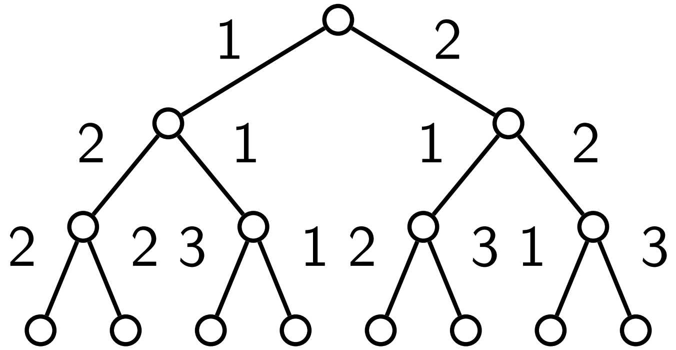
Explore nodes in increasing order of cost-so-far ($g(n)$):

# Uniform Cost Search (Dijkstra, 1959)



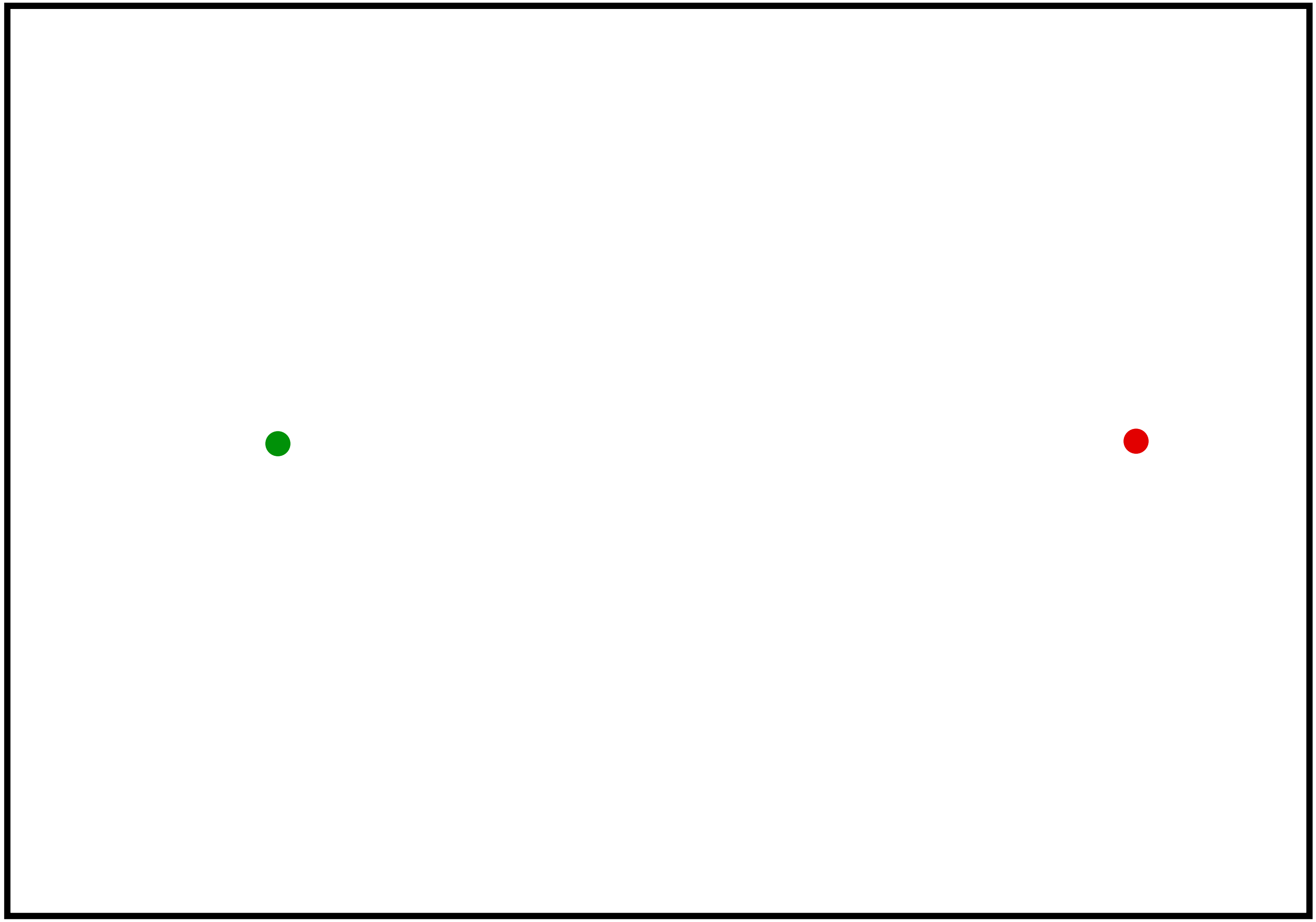
Explore nodes in increasing order of cost-so-far ($g(n)$):

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



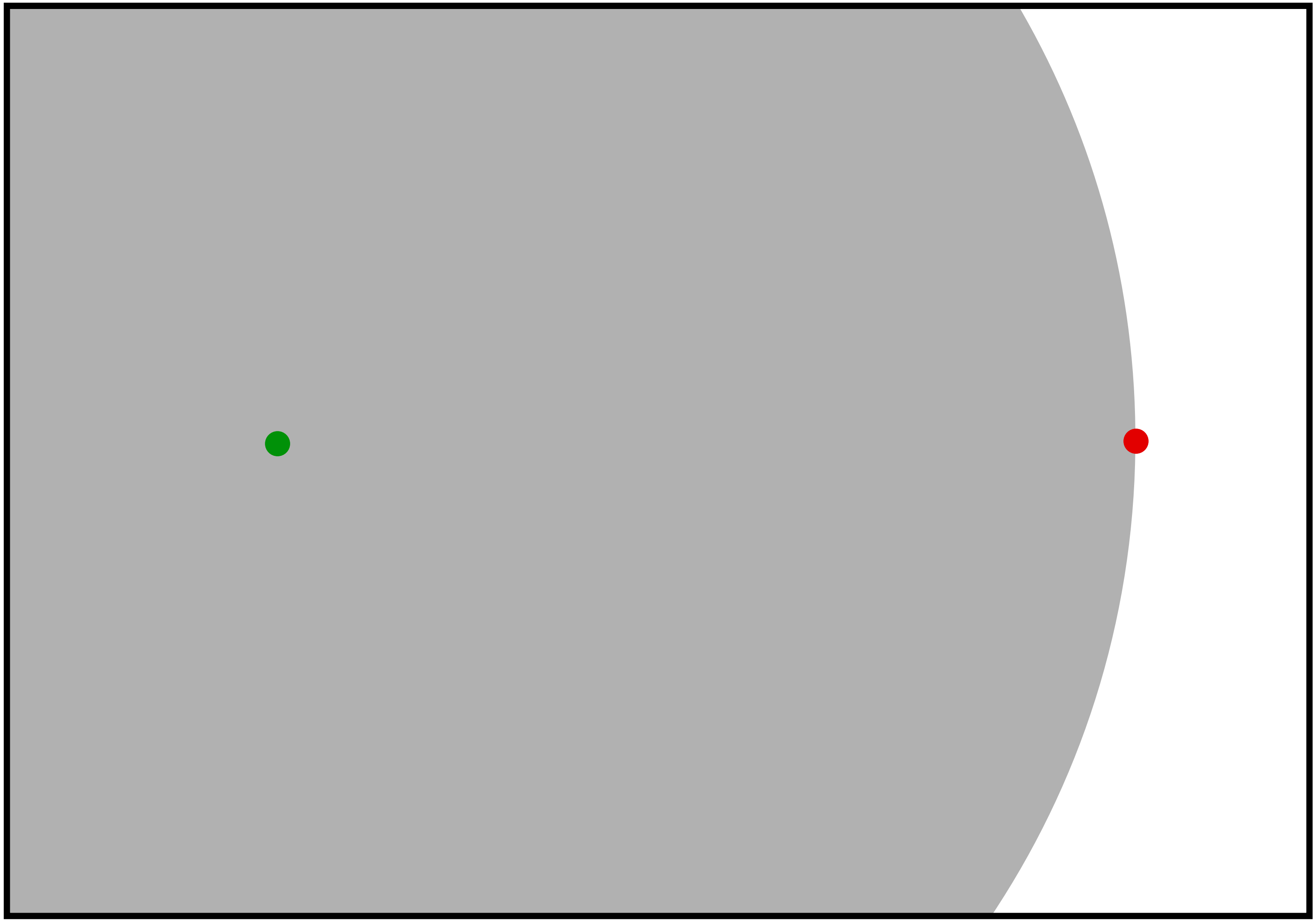
# Dijkstra Behavior





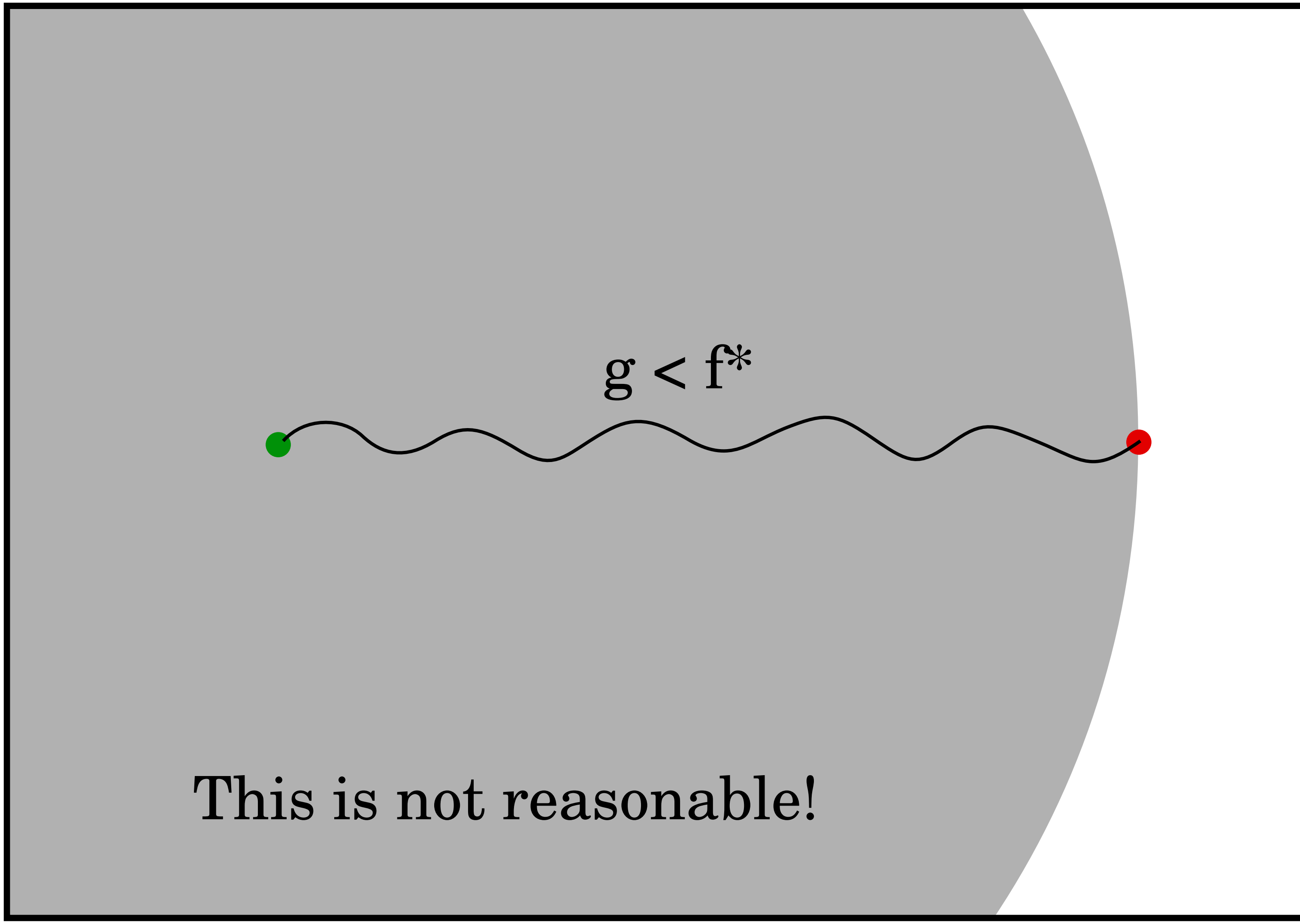
# Dijkstra Behavior





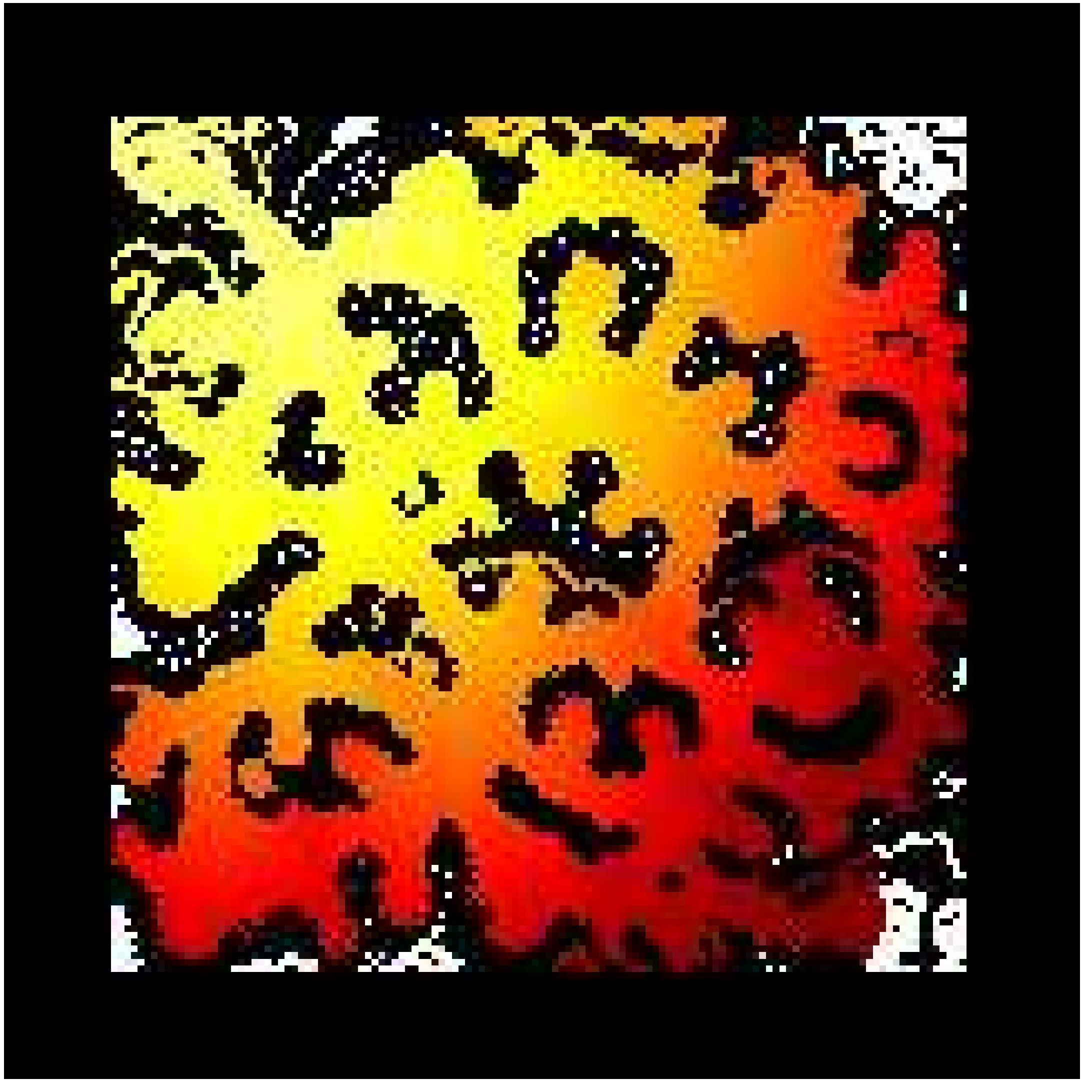
# Dijkstra Behavior





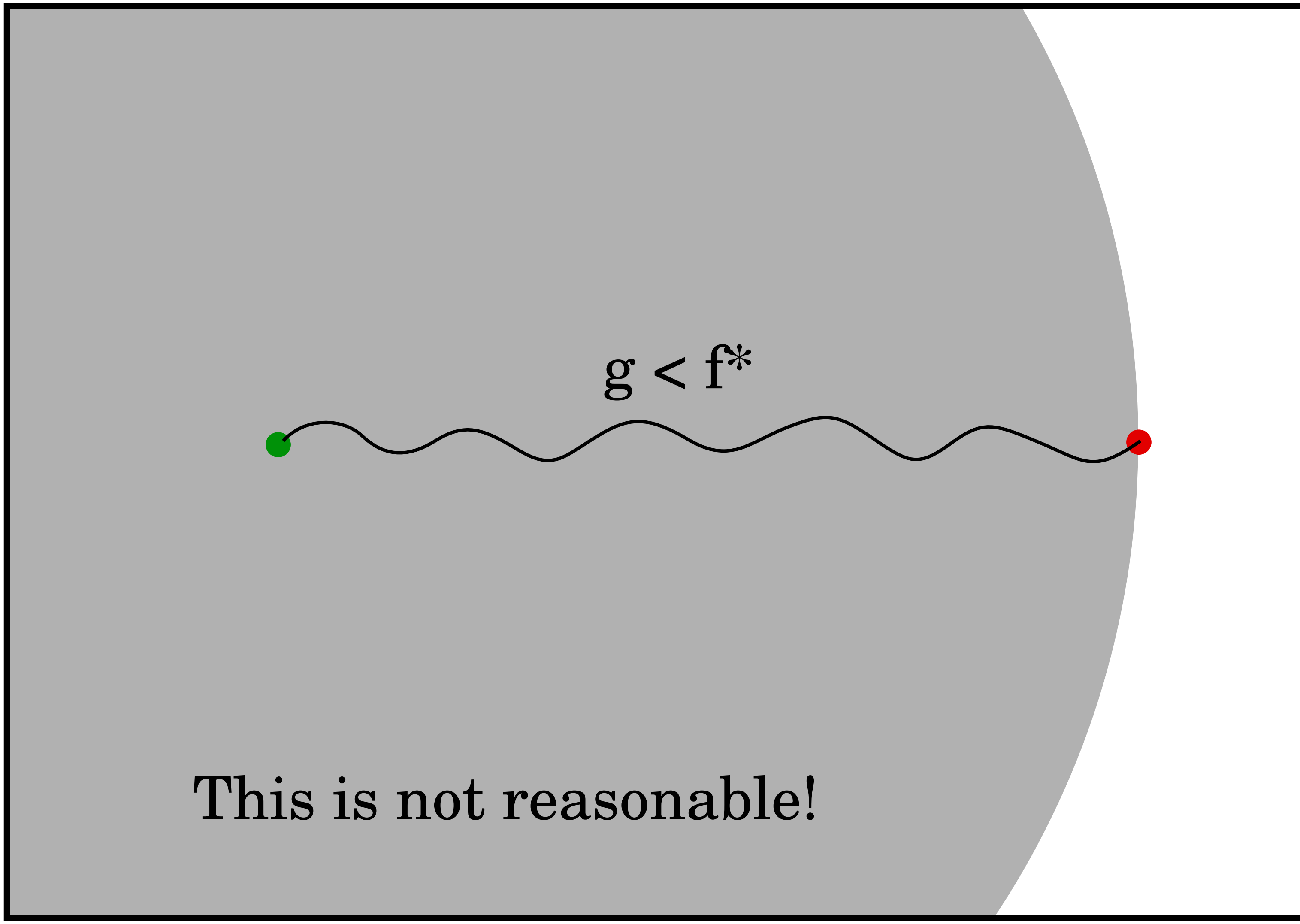
# Dijkstra: Pathfinding in Warcraft





# Problem-Specific Background Knowledge

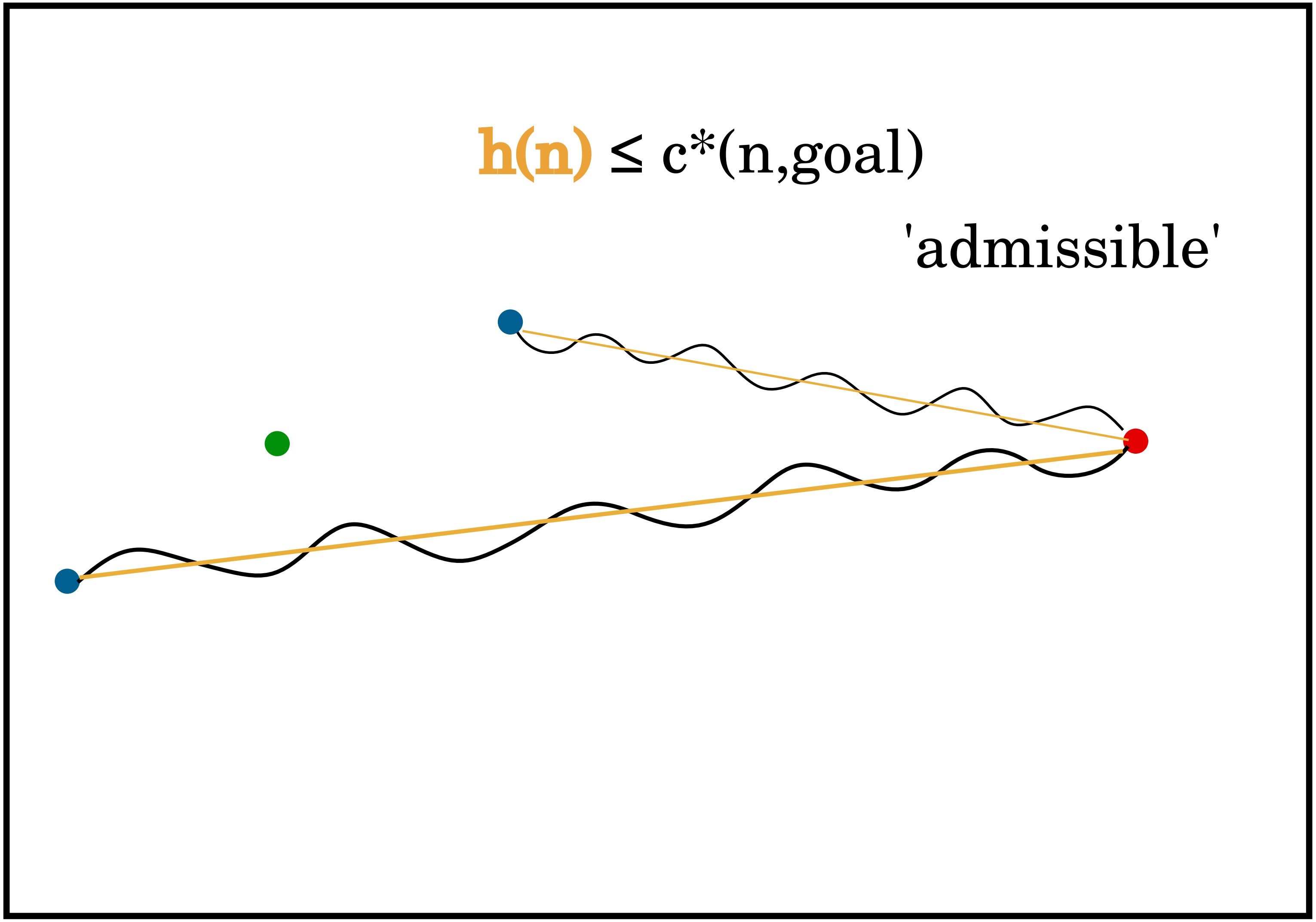





This is not reasonable!

# Heuristic Estimates of 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:

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

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



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

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

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

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



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

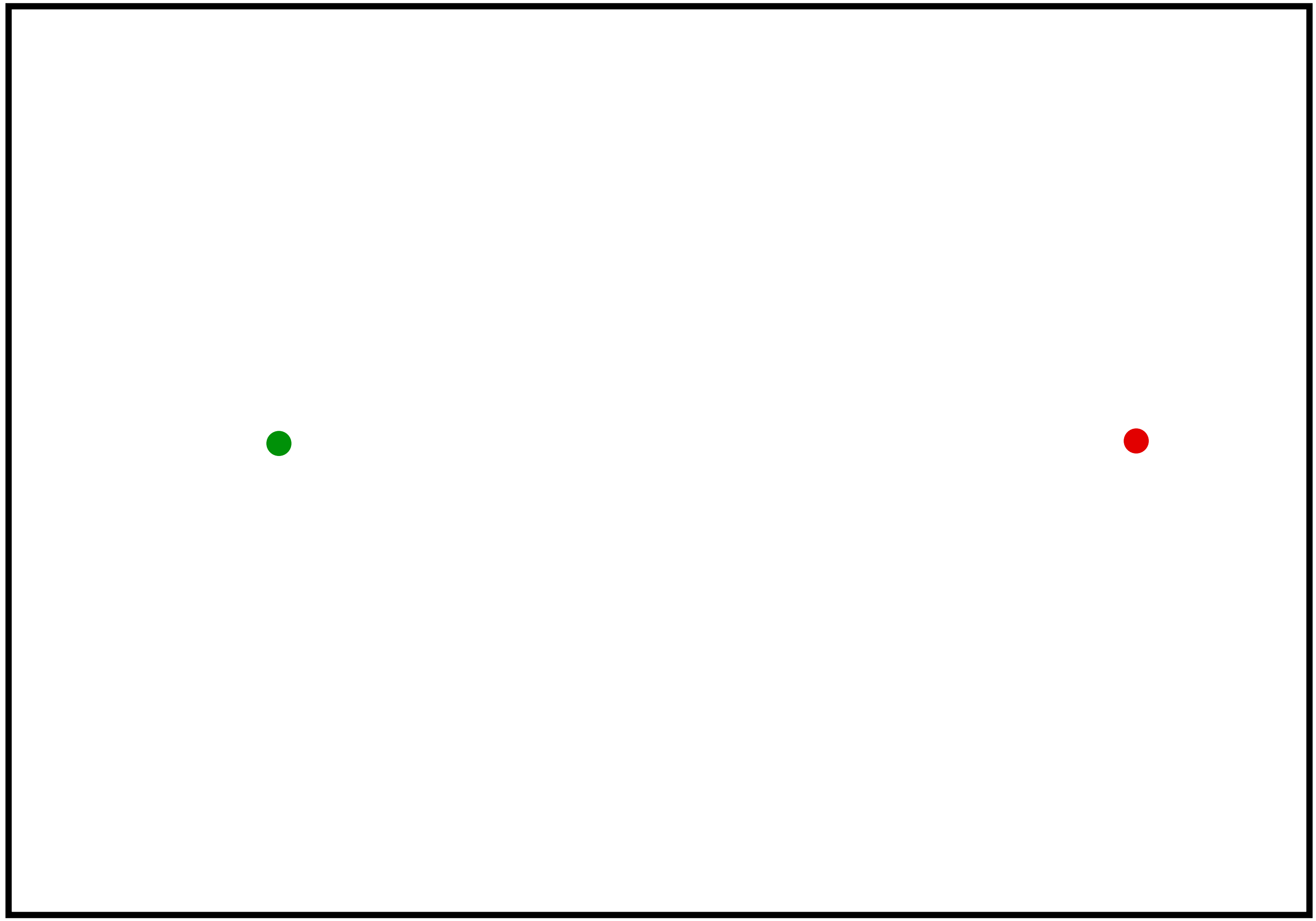
$g(n) =$ cost incurred so far
$h(n) =$ lower bound on cost to goal
$f(n) = g(n) + h(n)$

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

finds optimal solution if heuristic is admissible

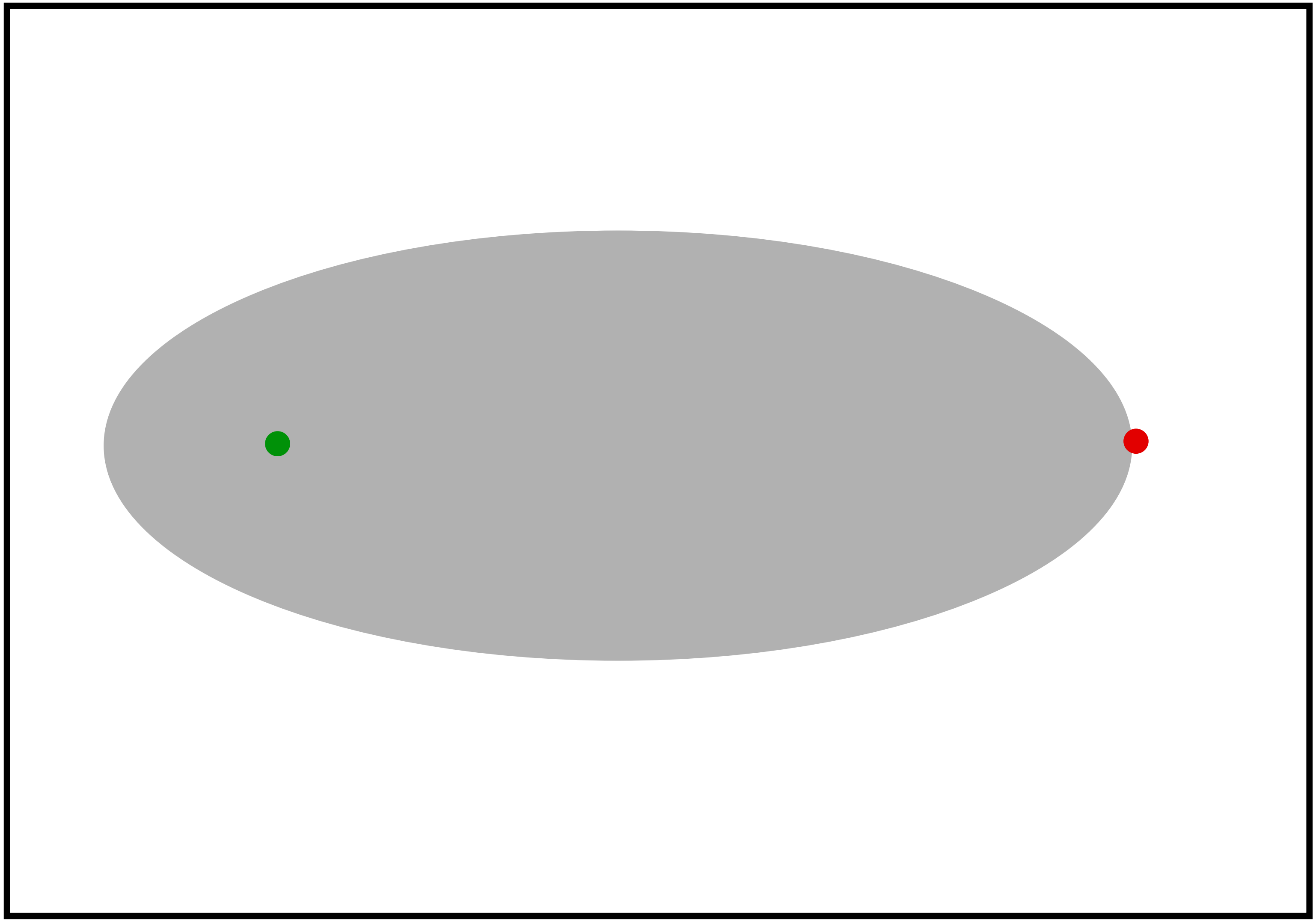
# A* Behavior





# A* Behavior

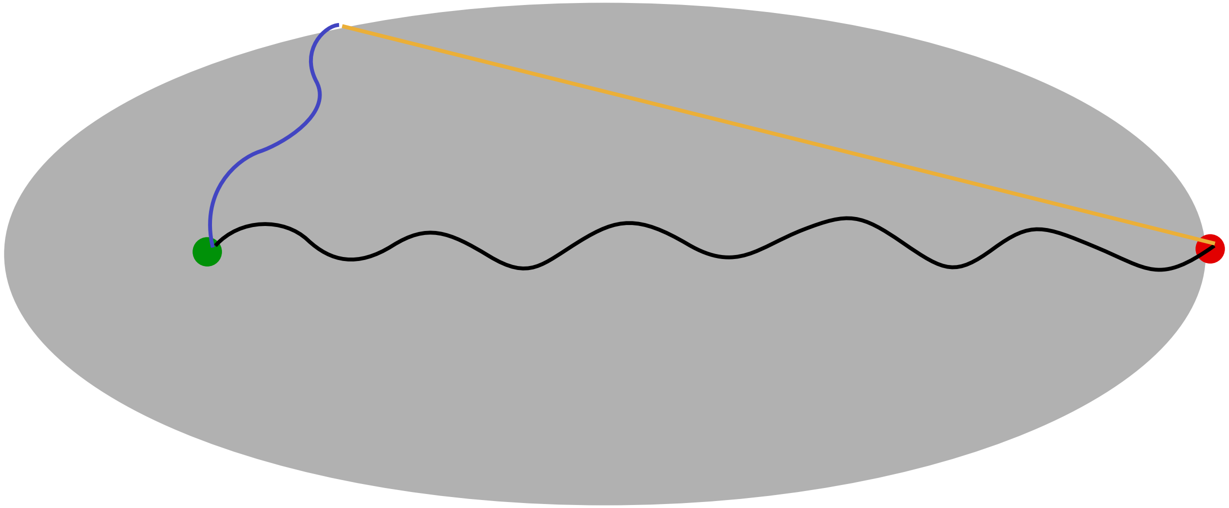




## A* Behavior

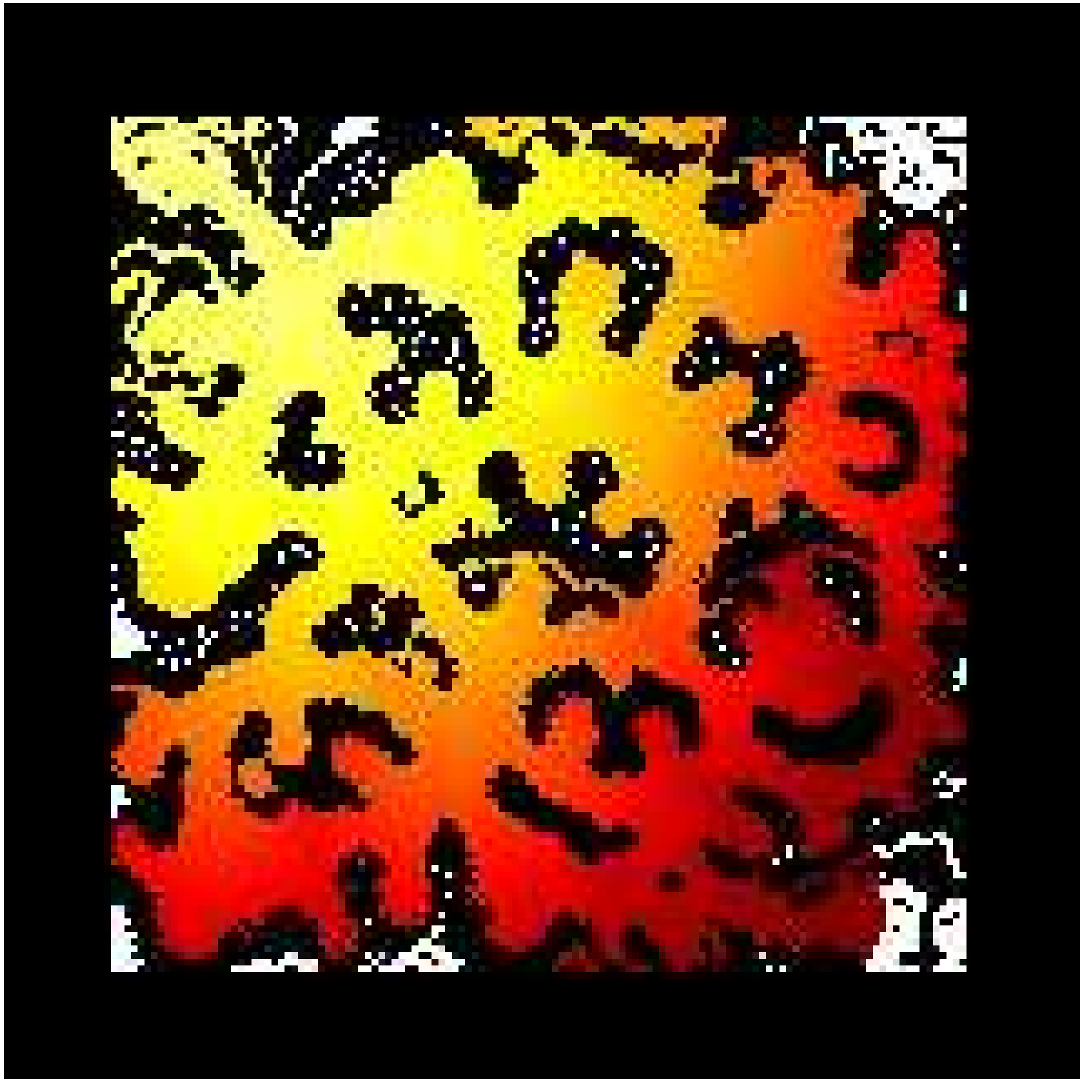


$$\mathbf{g} + \mathbf{h} < f^*$$



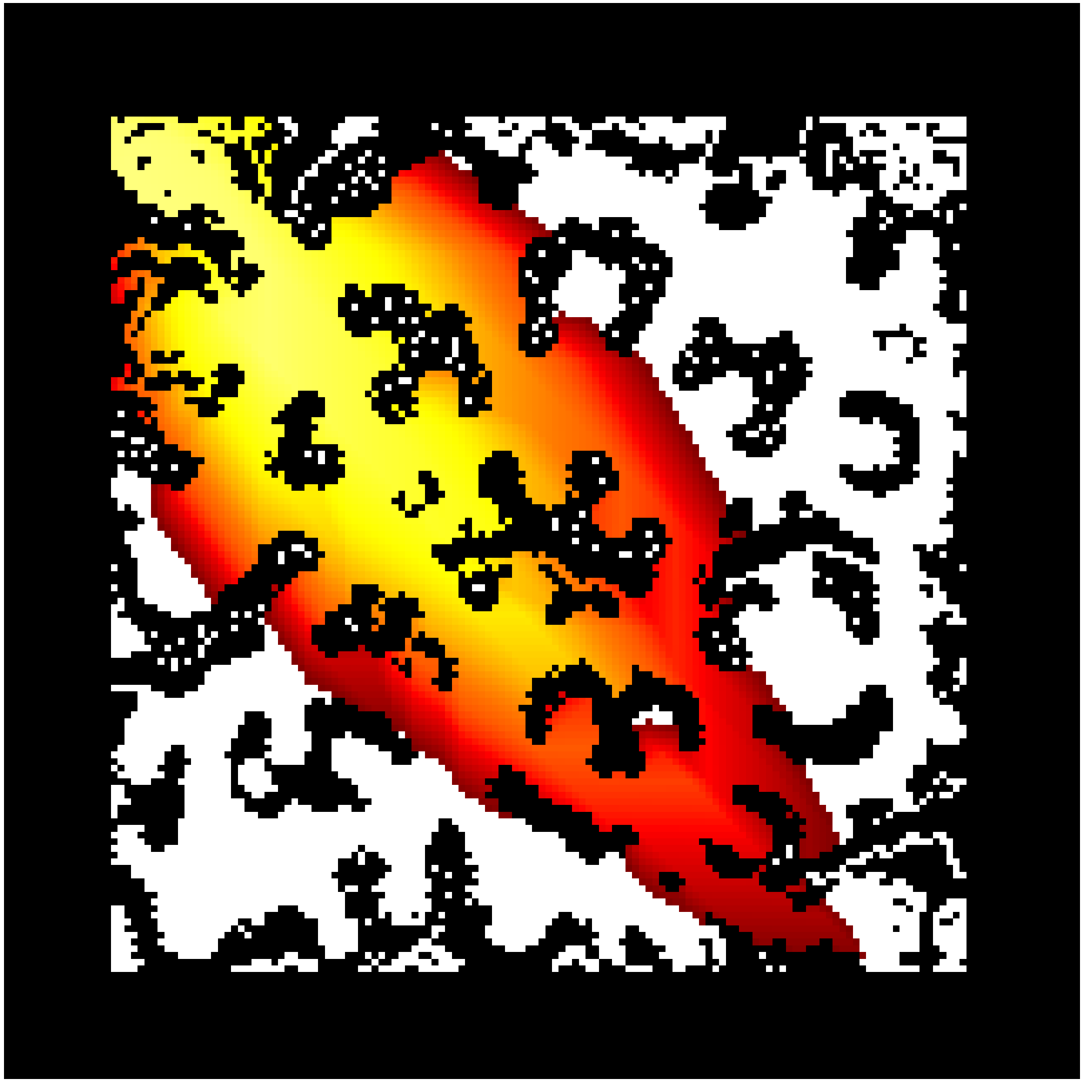
# Dijkstra vs A*: Pathfinding in Warcraft





# 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 cost of plan through $n$

# Quick Note: A Lower Bound on Solution Cost

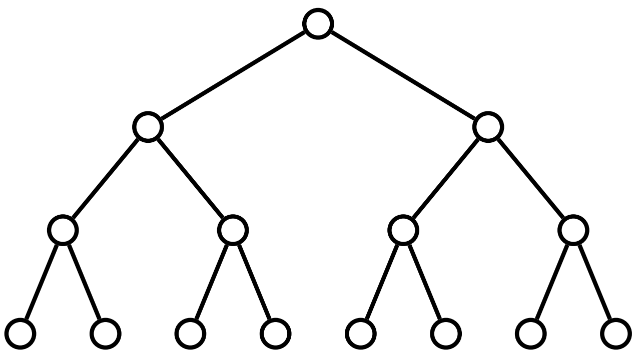


$f(n) = g(n) + h(n)$

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

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

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



# 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

Helmert and Röger, "How Good is Almost Perfect?" *AAAI-08*

# Problems with A*



A* takes exponential memory

Can sometimes be fixed: see 'iterative deepening'

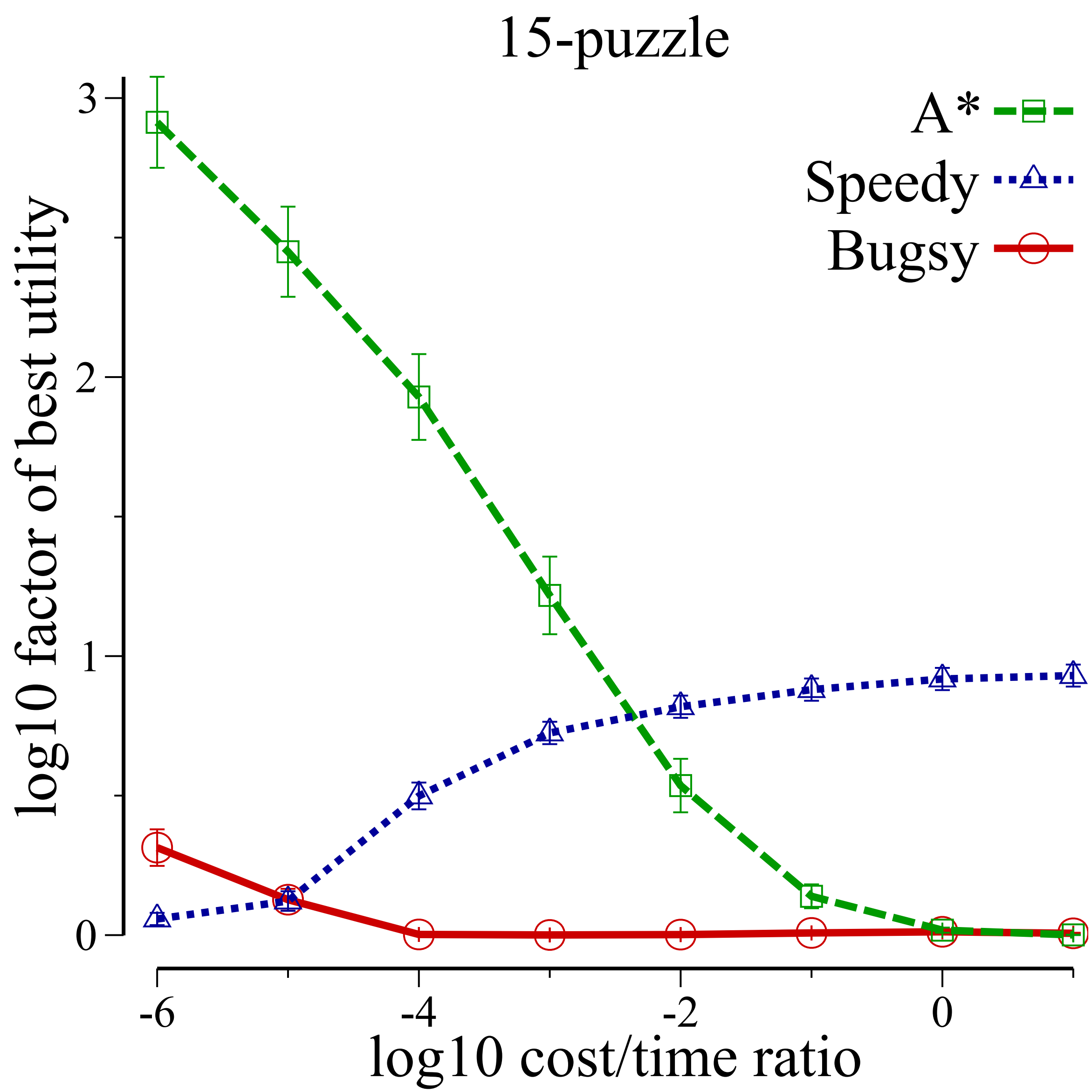
A* takes exponential time

Helmert and Röger, "How Good is Almost Perfect?" *AAAI-08*

We must trade cost for time.

# Optimizing Utility of Cost + 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

**anytime:** incrementally converge to optimal

**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

**anytime:** incrementally converge to optimal

**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

# Explicit Estimation Search (Thayer and Ruml, 2011)

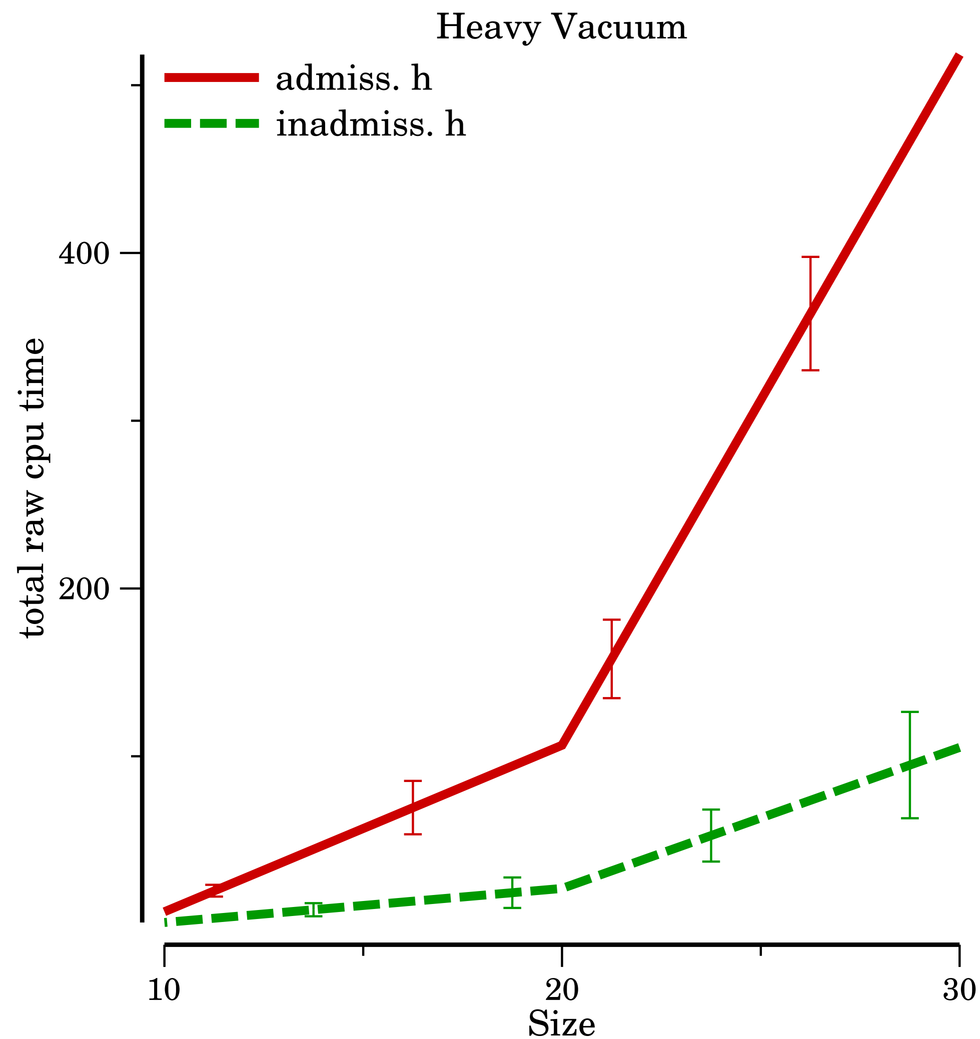


- unbiased estimates can be more informed than lower bounds
- nearest goal is the easiest to find

# Explicit Estimation Search (Thayer and Ruml, 2011)



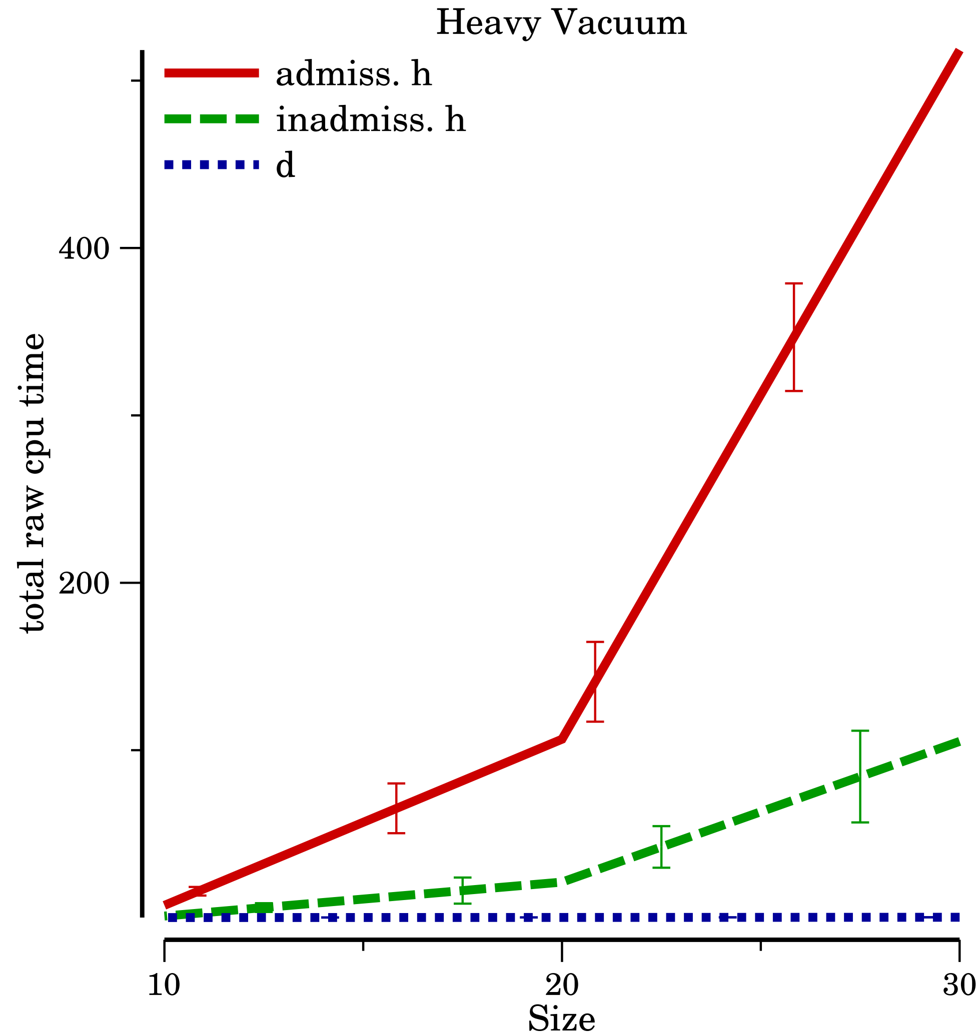
- unbiased estimates can be more informed than lower bounds
- nearest goal is the easiest to find



# Explicit Estimation Search (Thayer and Ruml, 2011)



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



# Explicit Estimation Search (Thayer and Ruml, 2011)



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

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

pursue nearest goal estimated to lie within 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 nearest goal estimated to lie within bound

# Search Strategy: Which Node to Expand?



$best_f$: open node giving lower bound on cost

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

# Search Strategy: Which Node to Expand?



$best_f$: open node giving lower bound on cost

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

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

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

# Search Strategy: Which Node to Expand?



$best_f$: open node giving lower bound on cost

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

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

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

pursue nearest goal estimated to lie within bound

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

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

# 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 nearest goal estimated to lie 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 nearest goal estimated to lie 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 nearest goal estimated to lie 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 nearest goal estimated to lie 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. Note: no magic numbers!

# 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



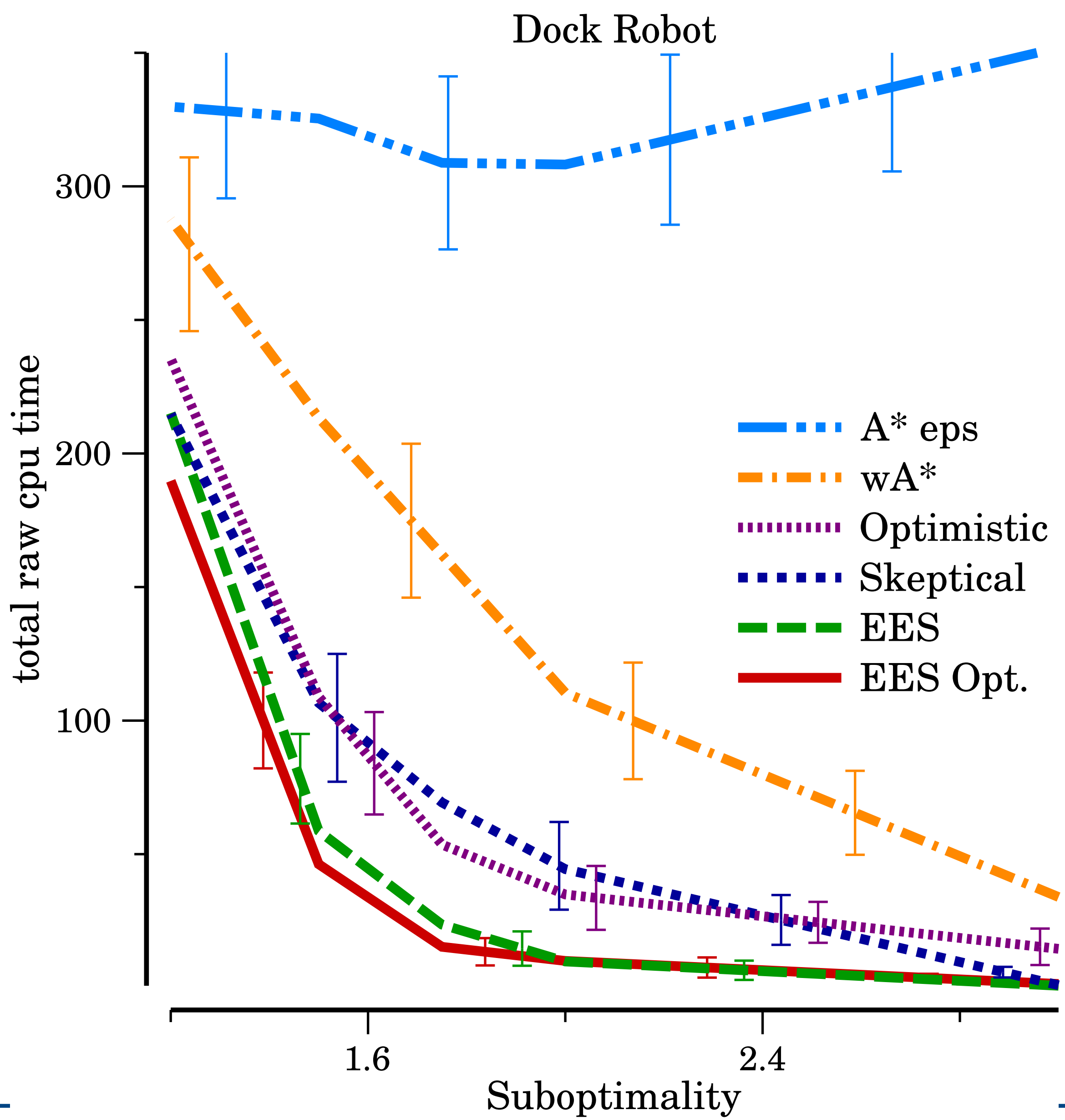
how does $\widehat{f}(n) \leq w \cdot f(best_f)$ ensure the suboptimality bound?

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

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

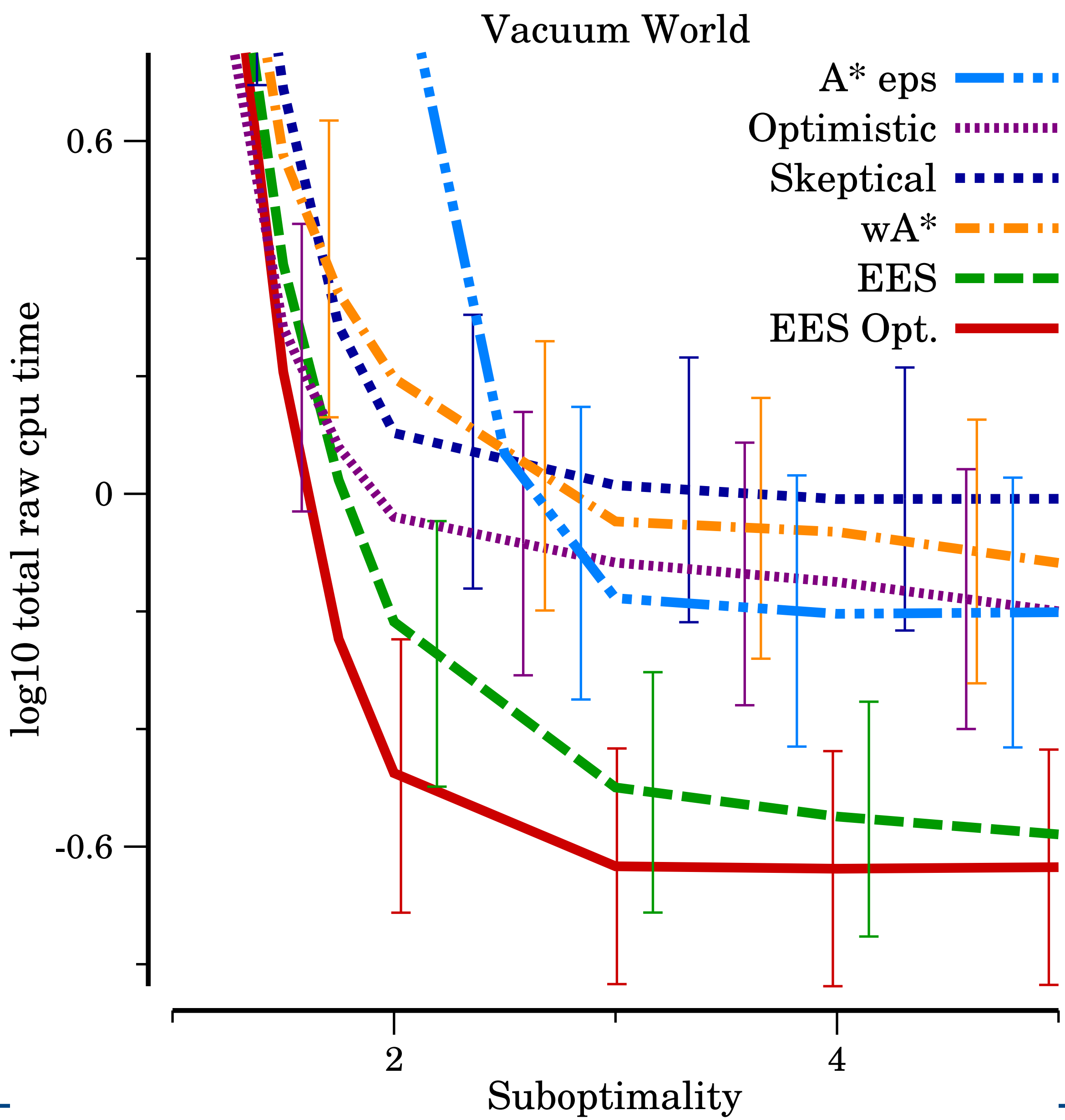
# EES Performance





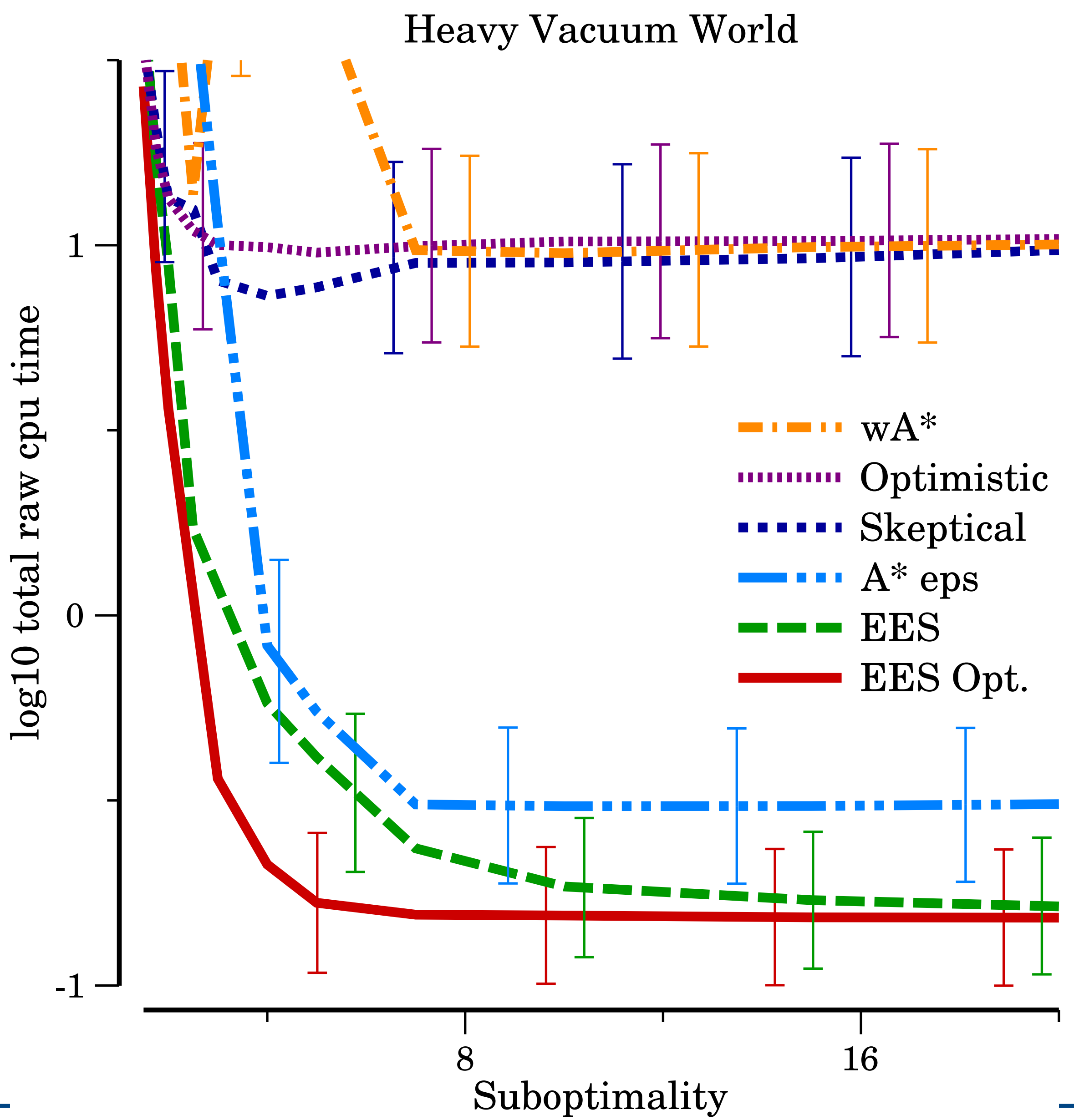
# EES Performance





# EES Performance





# Conclusion

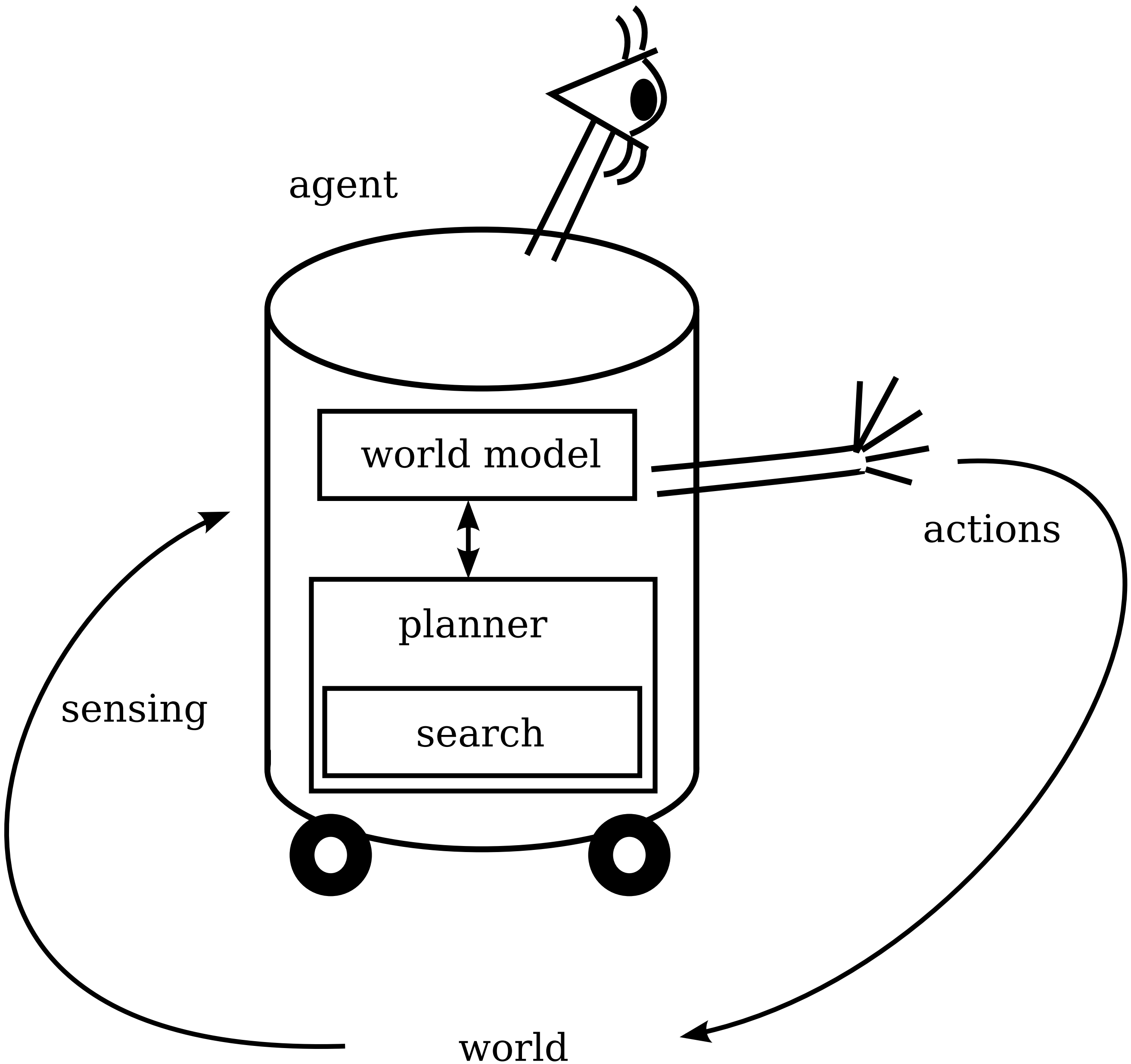
# A Science of Suboptimal Search



- what are the problem settings?
  - guarantees beyond optimal search
  - eg, bounded suboptimal search
  - utility-based optimization
- what are the sources of information?
  - where do inadmissible heuristics come from?
  - estimates instead of lower bounds
  - distance in addition to cost
- how to exploit and combine information?
  - new generation of suboptimal algorithms
  - meta-reasoning

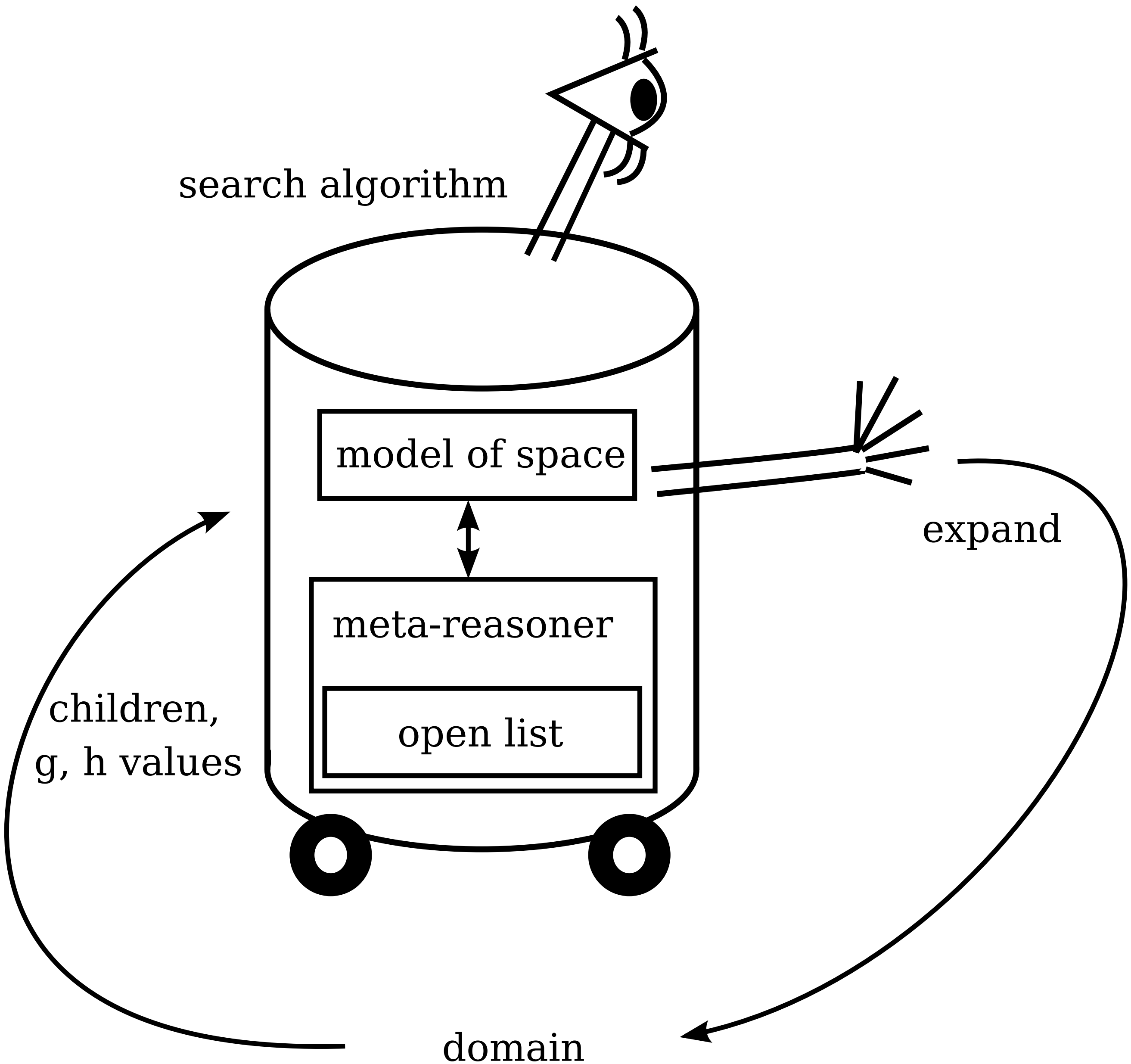
# The AI Vision





# Search Algorithms as Agents





The University of New Hampshire