Search Algorithms as Agents
— or —
There's More to Life Than $h(n)$

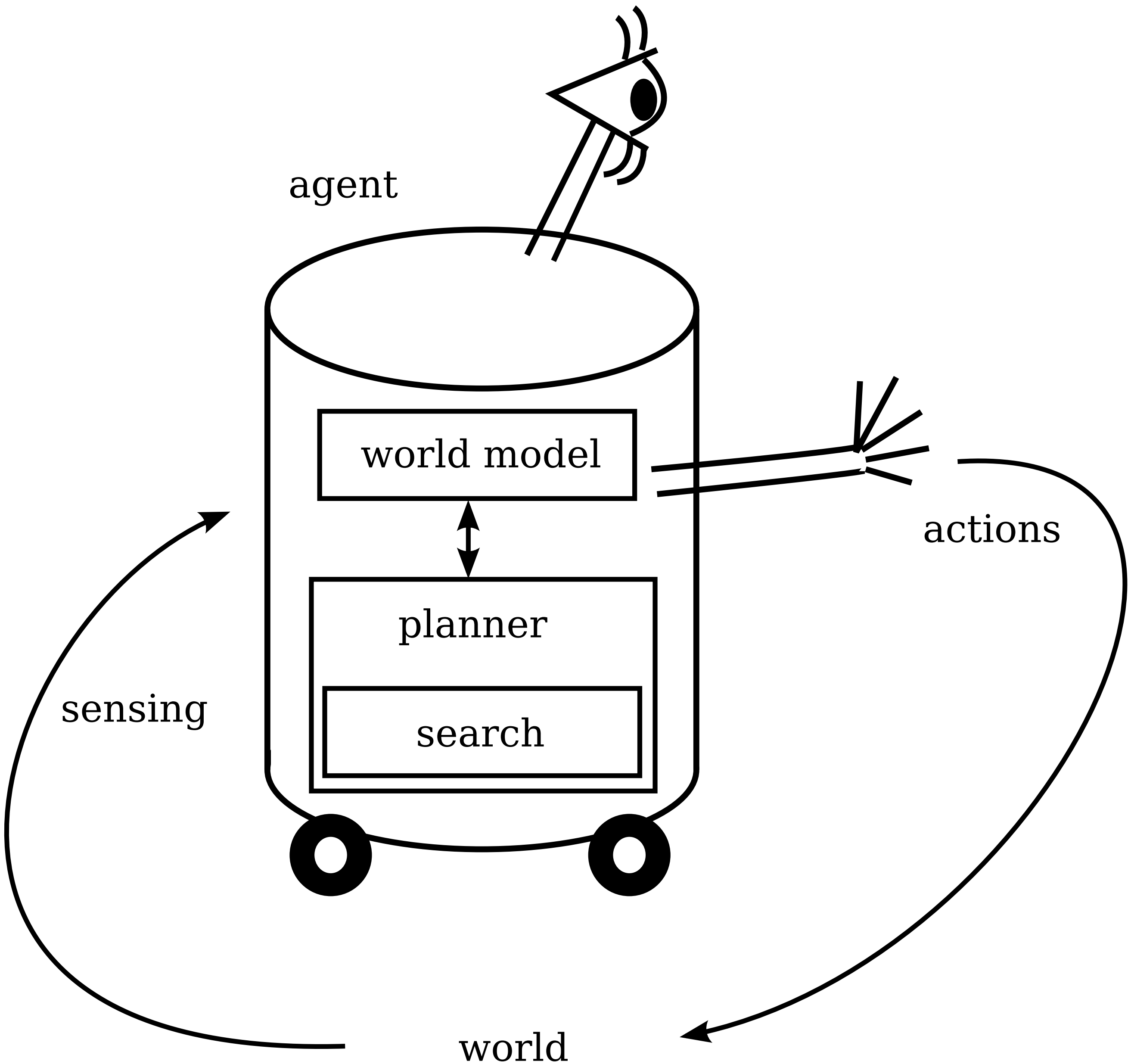
Wheeler Ruml

and the UNH AI Group, esp. Jordan Thayer

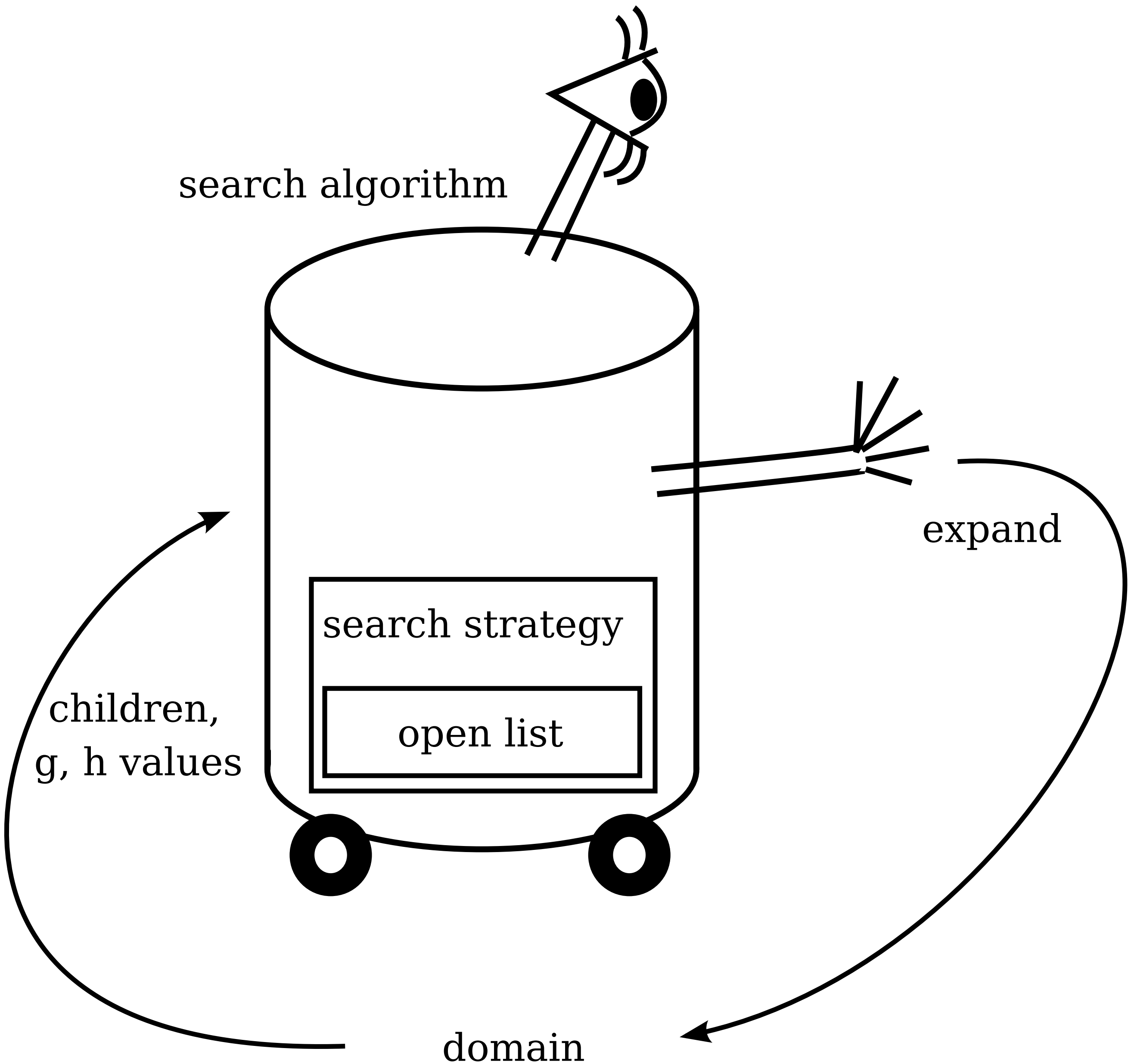
UNIVERSITY of NEW HAMPSHIRE

(thanks to the NSF RI and DARPA CSSG programs for support)

The AI Vision



Search Algorithms as Agents



What is Search?

acting under uncertainty to maximize utility

What is Search?

acting under uncertainty to maximize utility

= all of AI

What is Search?

acting under uncertainty to maximize utility
= all of AI

possible sources of information, ‘sensors’
how to exploit information, ‘aggregation’, ‘filtering’

Rational search:

- Othar Hansson
- Andy Mayer

Metareasoning, 'bounded optimality':

- Eric Wefald
- Shlomo Zilberstein
- Eric Horvitz
- Stuart Russell

Anytime search:

- Tom Dean
- Mark Boddy

optimal: minimize solution cost

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

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

Greedy Search

A Direct Approach

how to minimize solving time?

A Direct Approach

how to minimize solving time?

how to minimize number of expansions?

A Direct Approach

how to minimize solving time?

how to minimize number of expansions?

take the shortest path to a goal

A Direct Approach

how to minimize solving time?

how to minimize number of expansions?

take the shortest path to a goal

for domains with costs, this is not $h(n)$

A Direct Approach

how to minimize solving time?

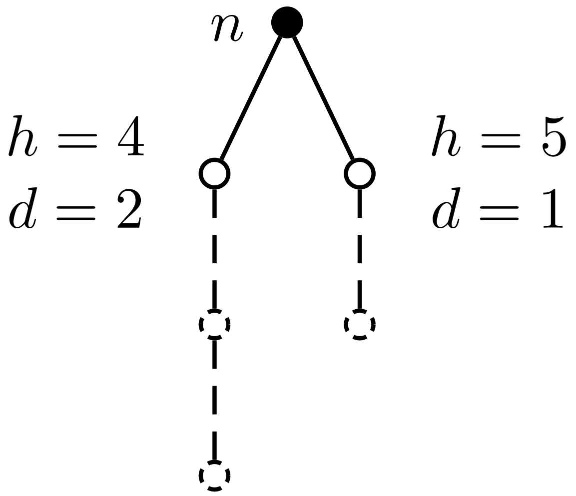
how to minimize number of expansions?

take the shortest path to a goal

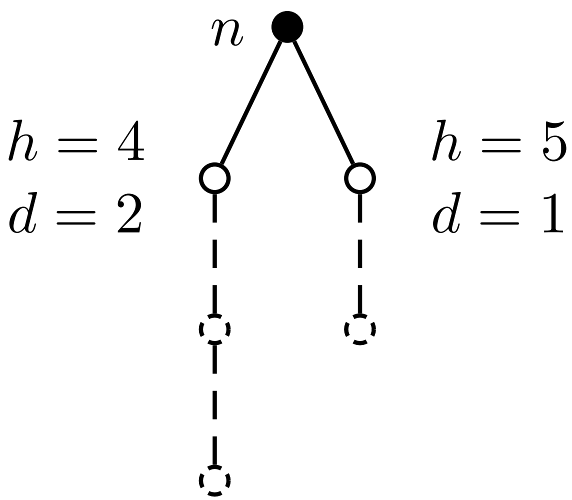
for domains with costs, this is not $h(n)$

source #1 of 5: distance-to-go

Distance-to-Go Estimates



Distance-to-Go Estimates



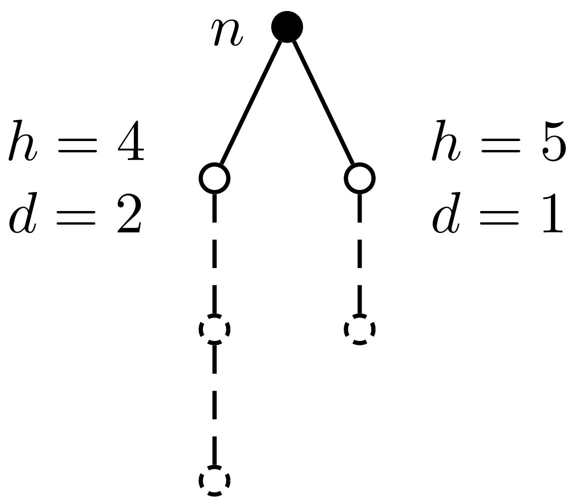
$$
\begin{aligned}
d^*_{cheapest}(n) &= 3 \\
d^*_{nearest}(n) &= 2
\end{aligned}
$$

$d_{nearest}$ is potentially independent of h,
can becomputed as h with unit costs

$d_{cheapest}$ often computable alongside h

d can be inadmissible

Distance-to-Go Estimates



$$d^*_{cheapest}(n) = 3$$
$$d^*_{nearest}(n) = 2$$

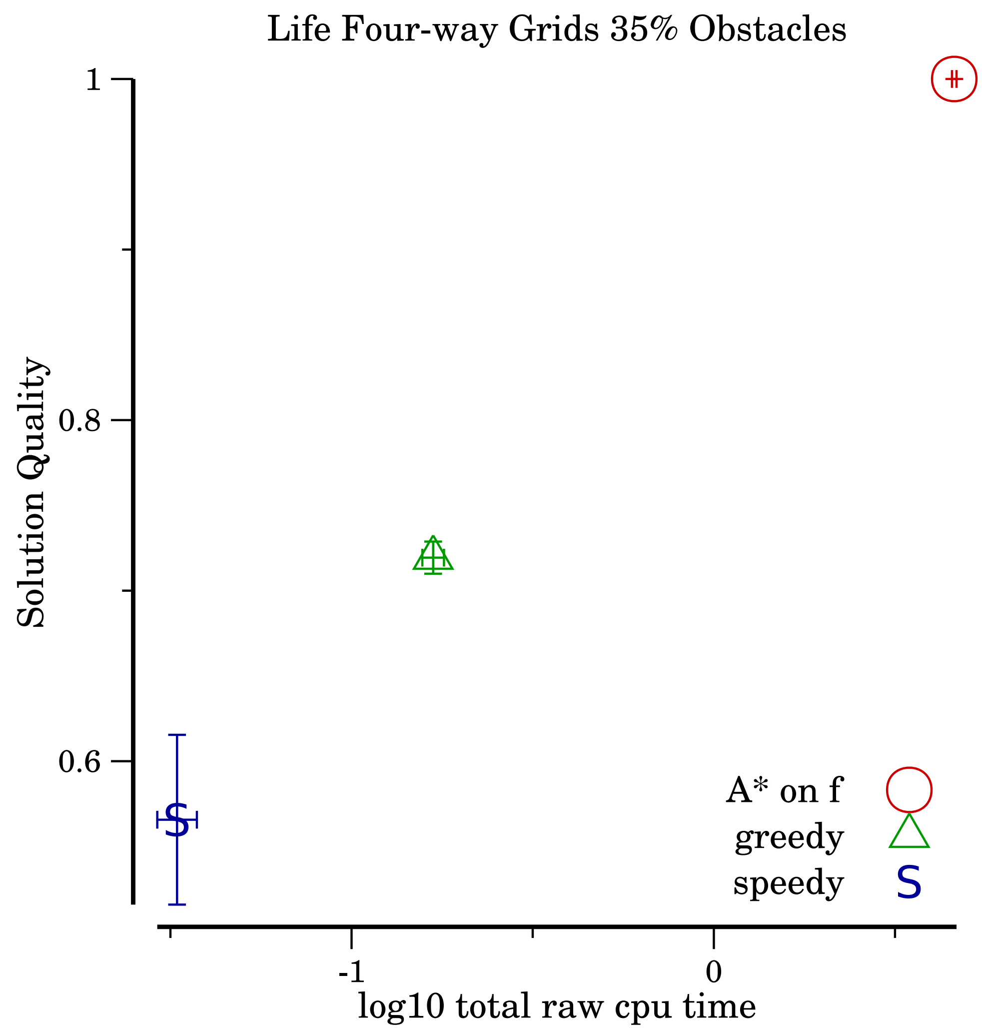
$d_{nearest}$ is potentially independent of h,
can becomputed as h with unit costs

$d_{cheapest}$ often computable alongside h

d can be inadmissible

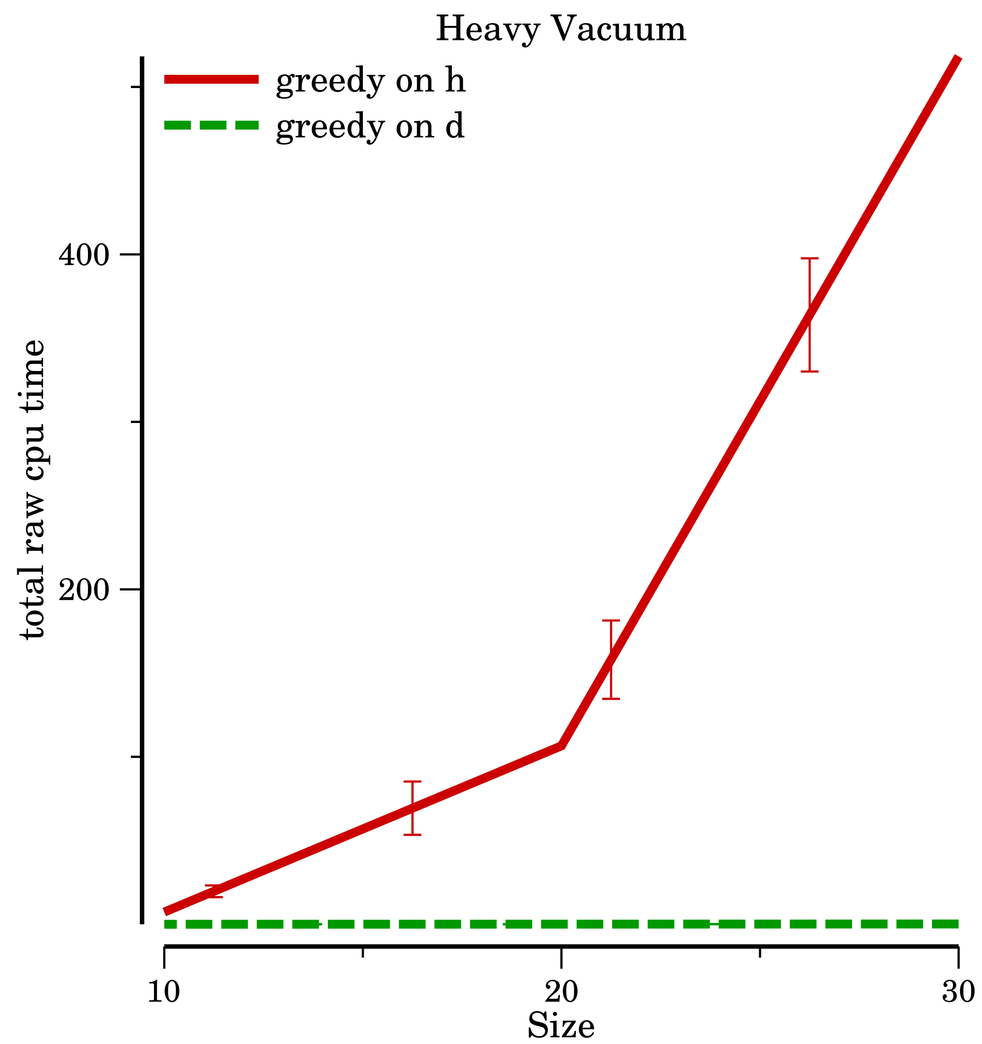
Speedy Search: best-first search on $d(n)$ (Thayer et al, SoCS-09)

Speedy Results



speedy finds solutions faster

Speedy Results



speedy scales better

Bounded-Suboptimal Search

A Direct Approach

minimize solving time subject to relative cost bound (factor of optimal)

A Direct Approach

minimize solving time subject to relative cost bound (factor of optimal)
weighted A* ($f'(n) = g(n) + w \cdot h(n)$) is simple but ad hoc

A Direct Approach

minimize solving time subject to relative cost bound (factor of optimal)
weighted A* ($f'(n) = g(n) + w \cdot h(n)$) is simple but ad hoc

expand the node closest to a solution within the bound

A Direct Approach

minimize solving time subject to relative cost bound (factor of optimal)
weighted A* ($f'(n) = g(n) + w \cdot h(n)$) is simple but ad hoc

expand the node closest to a solution within the bound
known to not work!

A^*_ϵ(Pearl and Kim, IEEE PAMI 1982)

intuition: of all solutions within the bound, the nearest
should be the fastest to find

A^*_ϵ (Pearl and Kim, IEEE PAMI 1982)

intuition: of all solutions within the bound, the nearest should be the fastest to find

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

$best_f$: open node with minimum f

two lists:

open: as usual, sorted on $f(n)$

focal: subset of open with $f(n) \leq w \cdot f(best_f)$, sorted on $d(n)$

A^*_ϵ: best-first search using *focal*

A^*_ϵ (Pearl and Kim, IEEE PAMI 1982)

intuition: of all solutions within the bound, the nearest should be the fastest to find

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

$best_f$: open node with minimum f

two lists:

open: as usual, sorted on $f(n)$

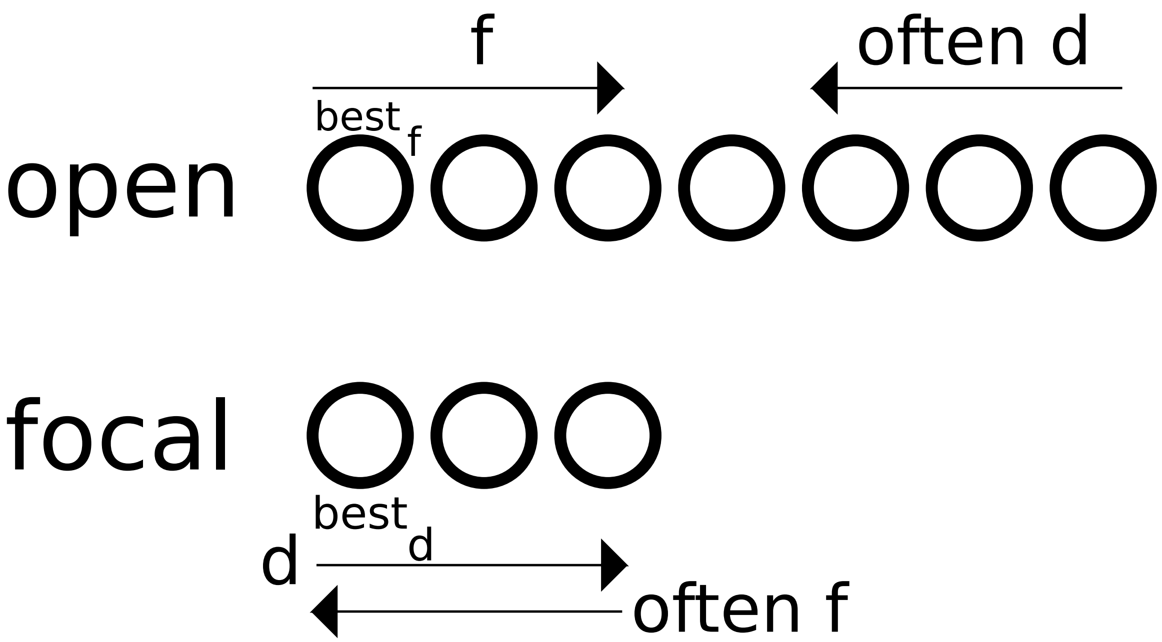
focal: subset of open with $f(n) \leq w \cdot f(best_f)$, sorted on $d(n)$

A^*_ϵ: best-first search using *focal*

Why doesn't it work?

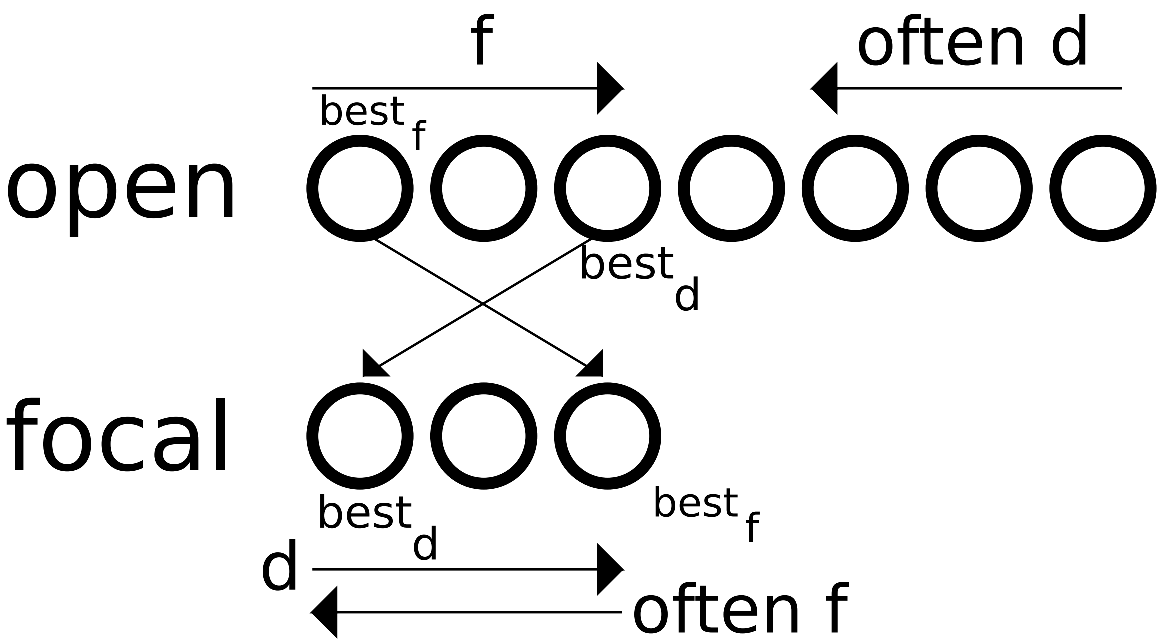
open: as usual, sorted on $f(n)$

focal: subset of open with $f(n) \leq w \cdot f(best_f)$, sorted on $d(n)$



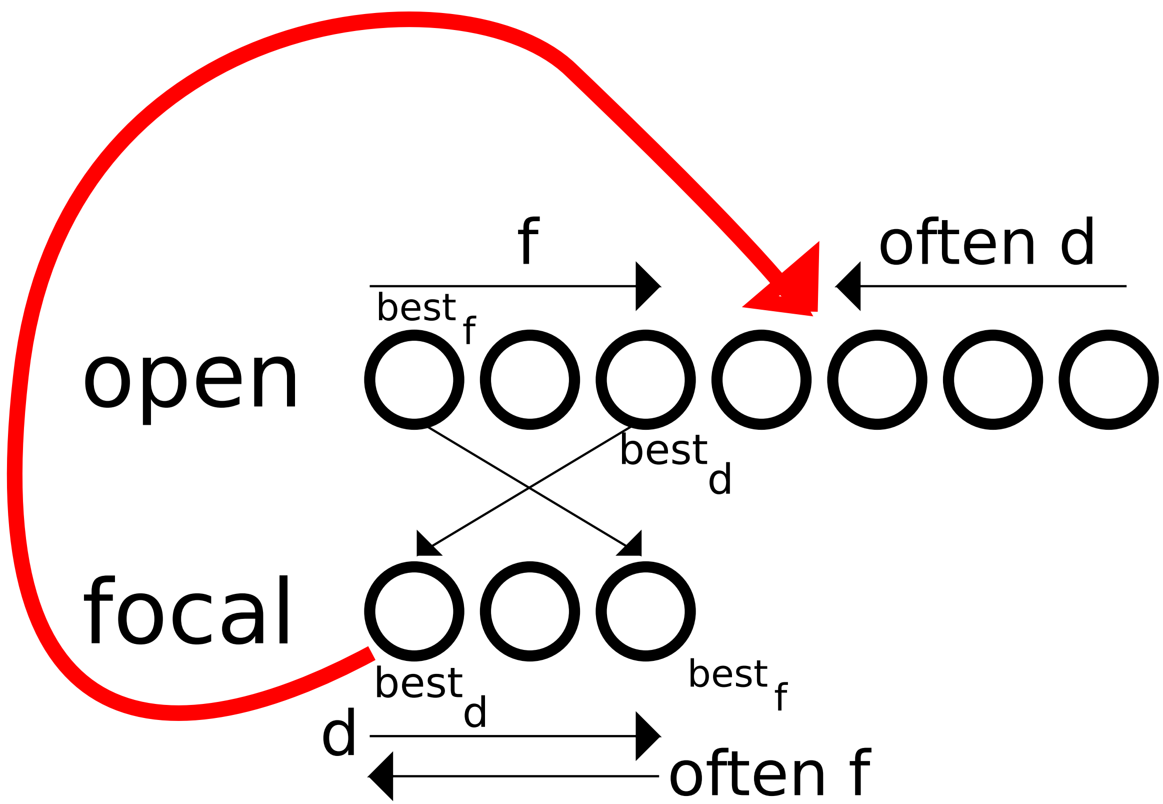
open: as usual, sorted on $f(n)$

focal: subset of open with $f(n) \le w \cdot f(best_f)$, sorted on $d(n)$



open: as usual, sorted on $f(n)$

focal: subset of open with $f(n) \leq w \cdot f(best_f)$, sorted on $d(n)$



f rises as search progresses (h is admissible)

$best_d$'s children don't qualify for *focal*

Estimated Cost

intuition: pursuing the shortest solution within the bound should be fast

intuition$'$: an unbiased estimates of cost won't always rise

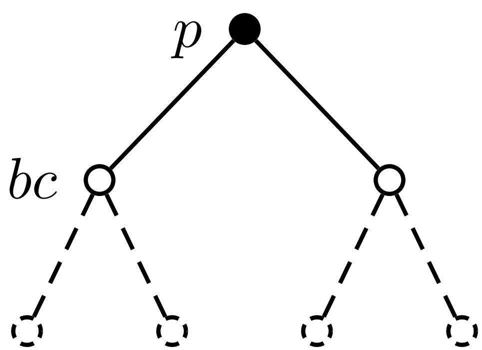
Estimated Cost

intuition: pursuing the shortest solution within the bound should be fast

intuition′: an unbiased estimates of cost won't always rise

source #2 of 5: unbiased cost estimates $\widehat{f}(n)$

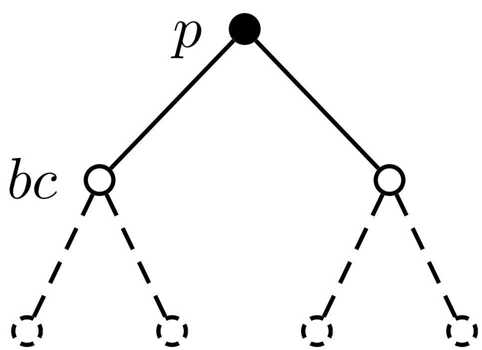
Every expansion gives evidence for heuristic's error!



$$
\begin{aligned}
f^*(p) &= f^*(bc) \\
h^*(p) &= h^*(bc) + c(p, bc) \\
\epsilon_h &= (h(bc) + c(p, bc)) - h(p)
\end{aligned}
$$

Learning Unbiased Cost Estimates

Every expansion gives evidence for heuristic's error!

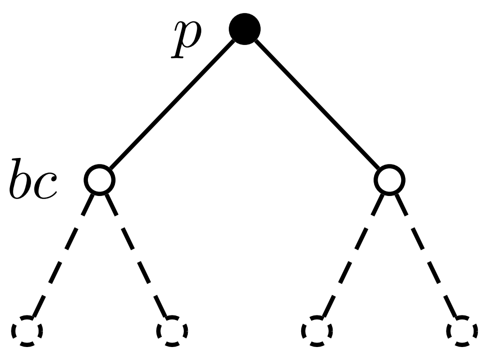


$$\begin{aligned} f^*(p) &= f^*(bc) \\ h^*(p) &= h^*(bc) + c(p, bc) \\ \epsilon_h &= (h(bc) + c(p, bc)) - h(p) \end{aligned}$$

$$\widehat{h}(n) = h(n) + \overline{\epsilon_h} \cdot d(n)$$

Learning Unbiased Cost Estimates

Every expansion gives evidence for heuristic's error!



$$\begin{aligned} f^*(p) &= f^*(bc) \\ h^*(p) &= h^*(bc) + c(p, bc) \\ \epsilon_h &= (h(bc) + c(p, bc)) - h(p) \end{aligned}$$

$$\widehat{h}(n) = h(n) + \overline{\epsilon_h} \cdot d(n)$$

can do this for $d(n)$ too...

$$\widehat{h}(n) = h(n) + \overline{\epsilon_h} \cdot \widehat{d}(n)$$

see Thayer et al (ICAPS-11) for details

Explicit Estimation Search (EES, IJCAI-11)

$best_f$: open node with minimum f

$best_{\widehat{f}}$: open node with minimum $\widehat{f}$

Explicit Estimation Search (EES, IJCAI-11)

$best_f$: open node with minimum f

$best_{\widehat{f}}$: open node with minimum $\widehat{f}$

three lists (!):

open: as usual, but sorted on $\widehat{f}(n)$

focal: subset of open with $\widehat{f}(n) \leq w \cdot \widehat{f}(best_{\widehat{f}})$, sorted on $\widehat{d}(n)$

cleanup: same as *open*, but sorted on $f(n)$

Explicit Estimation Search (EES, IJCAI-11)

$best_f$: open node with minimum f

$best_{\widehat{f}}$: open node with minimum $\widehat{f}$

three lists (!):

- *open*: as usual, but sorted on $\widehat{f}(n)$
- *focal*: subset of open with $\widehat{f}(n) \leq w \cdot \widehat{f}(best_{\widehat{f}})$, sorted on $\widehat{d}(n)$
- *cleanup*: same as *open*, but sorted on $f(n)$

$best_{\widehat{d}}$: first node on *focal*

Explicit Estimation Search (EES, IJCAI-11)

$best_f$: open node with minimum f

$best_{\widehat{f}}$: open node with minimum $\widehat{f}$

three lists (!):

open: as usual, but sorted on $\widehat{f}(n)$

focal: subset of open with $\widehat{f}(n) \leq w \cdot \widehat{f}(best_{\widehat{f}})$, sorted on $\widehat{d}(n)$

cleanup: same as *open*, but sorted on $f(n)$

$best_{\widehat{d}}$: first node on *focal*

estimated w-admissible node with minimum $\widehat{d}$

EES Expansion Order

$best_f$: open node with minimum f
$best_{\widehat{f}}$: open node with minimum $\widehat{f}$
$best_{\widehat{d}}$: estimated w-admissible node with minimum $\widehat{d}$

node to expand next:

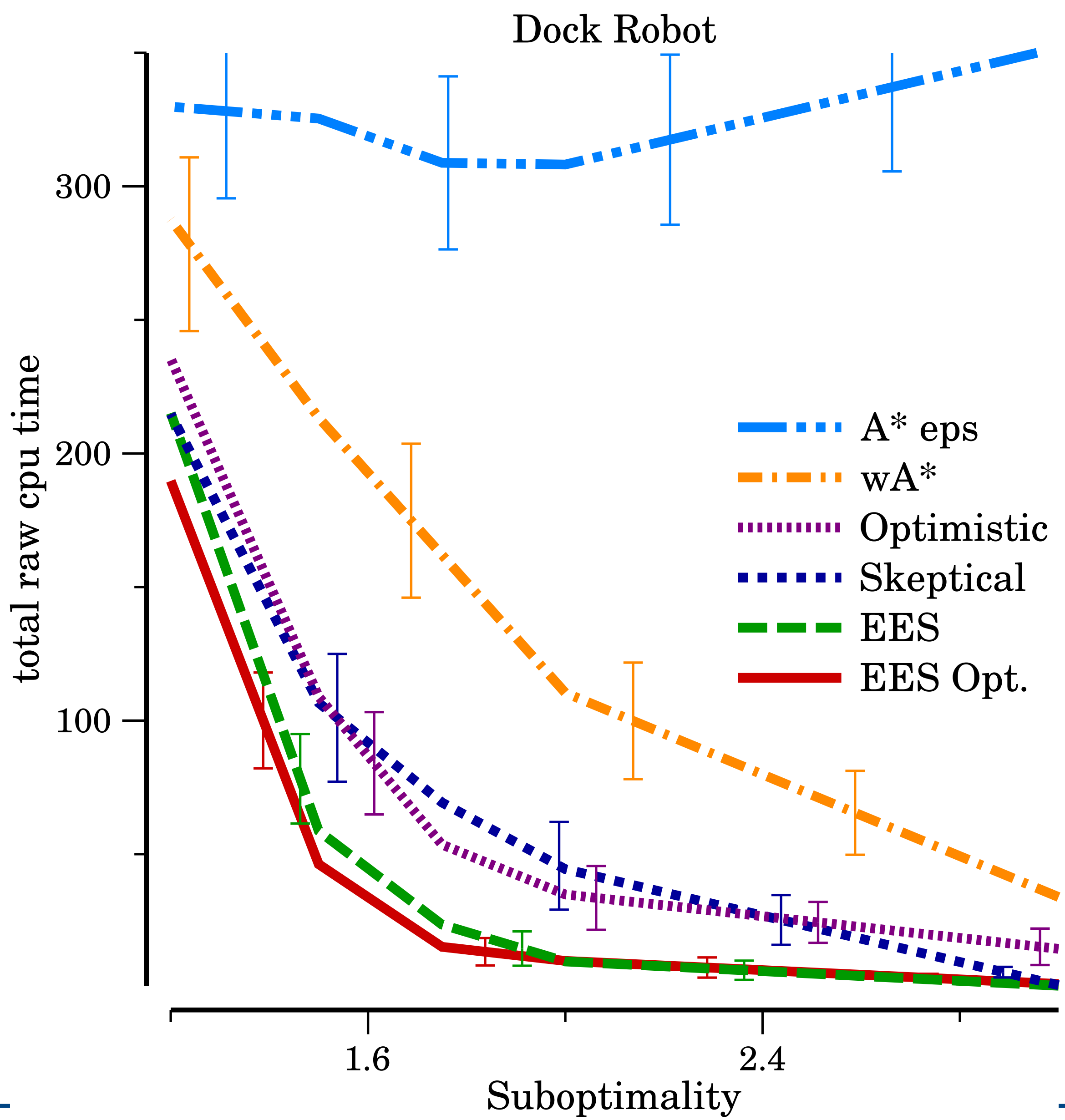
1. pursue the shortest solution within the bound
2. pursue the estimated cheapest solution
3. raise the lower bound on optimal cost

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** $best_f$

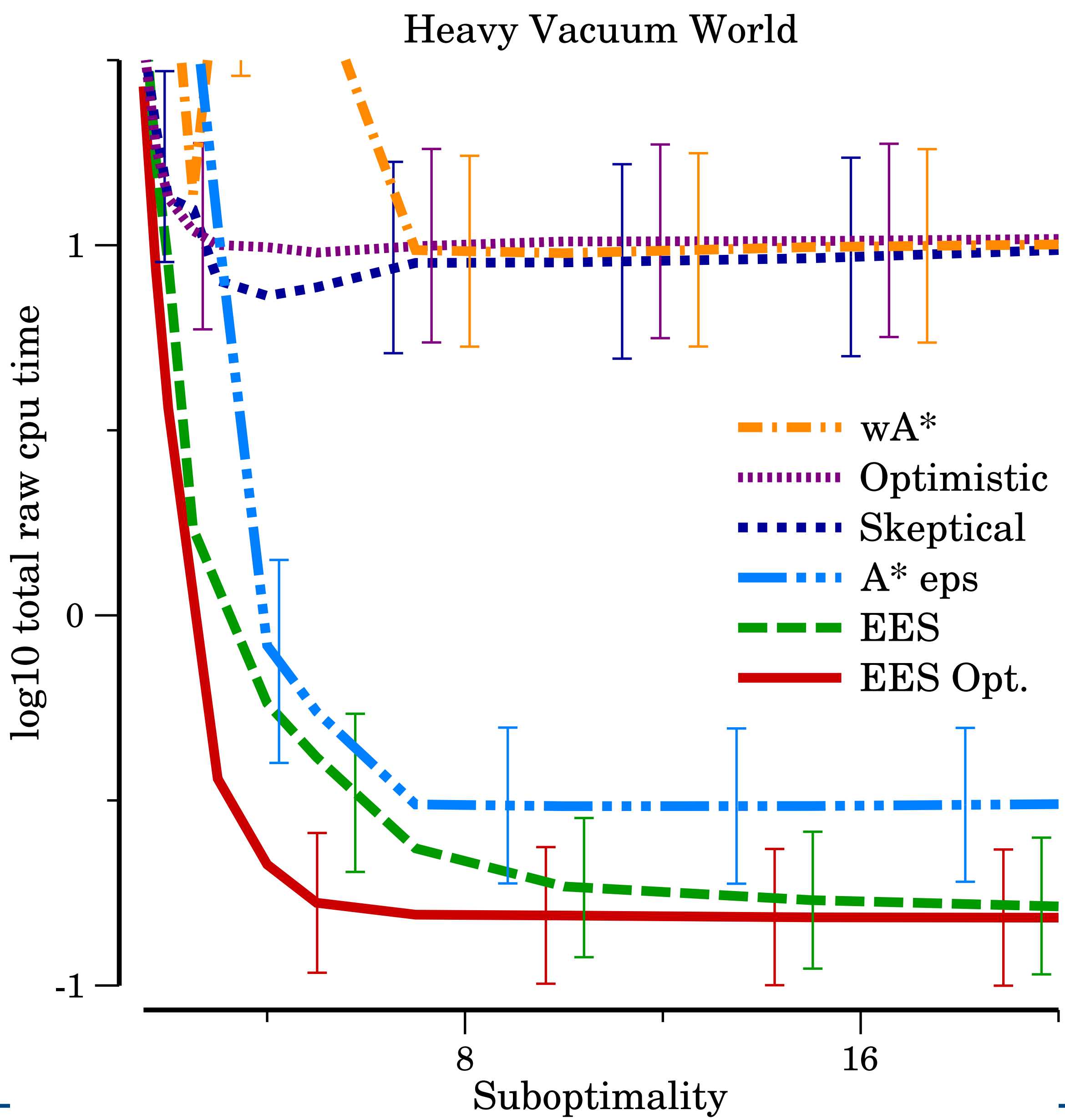
find a solution provably within the bound as quickly as possible

EES Results



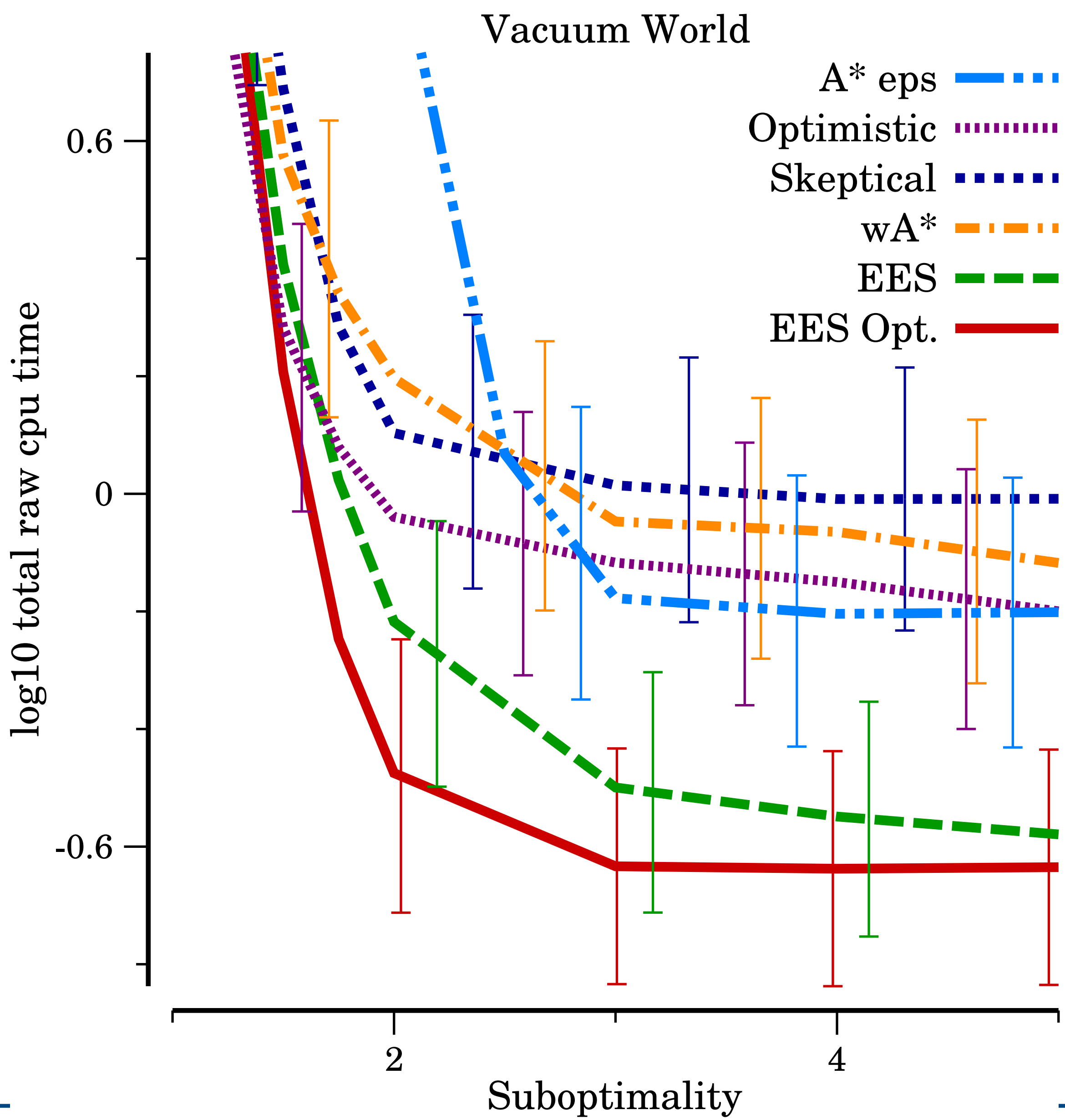
Dock Robot
total raw cpu time
300
200
100
1.6
2.4
Suboptimality
A* eps
wA*
Optimistic
Skeptical
EES
EES Opt.

EES Results



Heavy Vacuum World
log10 total raw cpu time
1
0
-1
8
16
Suboptimality
wA*
Optimistic
Skeptical
A* eps
EES
EES Opt.

EES Results



Vacuum World
A* eps
Optimistic
Skeptical
wA*
EES
EES Opt.
log10 total raw cpu time
0.6
0
-0.6
2
4
Suboptimality

The Story So Far

- search algorithms as agents
- more information sources
  1. $d(n)$: distance-to-go
  2. $\widehat{f}(n)$: expected cost (expansion experience)
- more problem settings
  - greedy search: Speedy
  - bounded-suboptimal search: EES

Contract Search

A Direct Approach

minimize cost subject to absolute time bound

A Direct Approach

minimize cost subject to absolute time bound

anytime algorithms: series of solutions, eg, unknown deadline

A Direct Approach

minimize cost subject to absolute time bound
anytime algorithms: series of solutions, eg, unknown deadline

while time remains
    expand node with best reachable solution

A Direct Approach

minimize cost subject to absolute time bound

anytime algorithms: series of solutions, eg, unknown deadline

while time remains
    expand node with best reachable solution

best: expected cost

reachable: expected search time to goal

Vacillation

how long to reach goal $\widehat{d}(n)$ steps away?

Vacillation

how long to reach goal $\widehat{d}(n)$ steps away?

how long to go one step toward a goal?

estimate $\overline{\Delta e}$, time between a node's generation and expansion

Vacillation

how long to reach goal $\widehat{d}(n)$ steps away?

how long to go one step toward a goal?

estimate $\overline{\Delta e}$, time between a node's generation and expansion

just record current time in node when generating

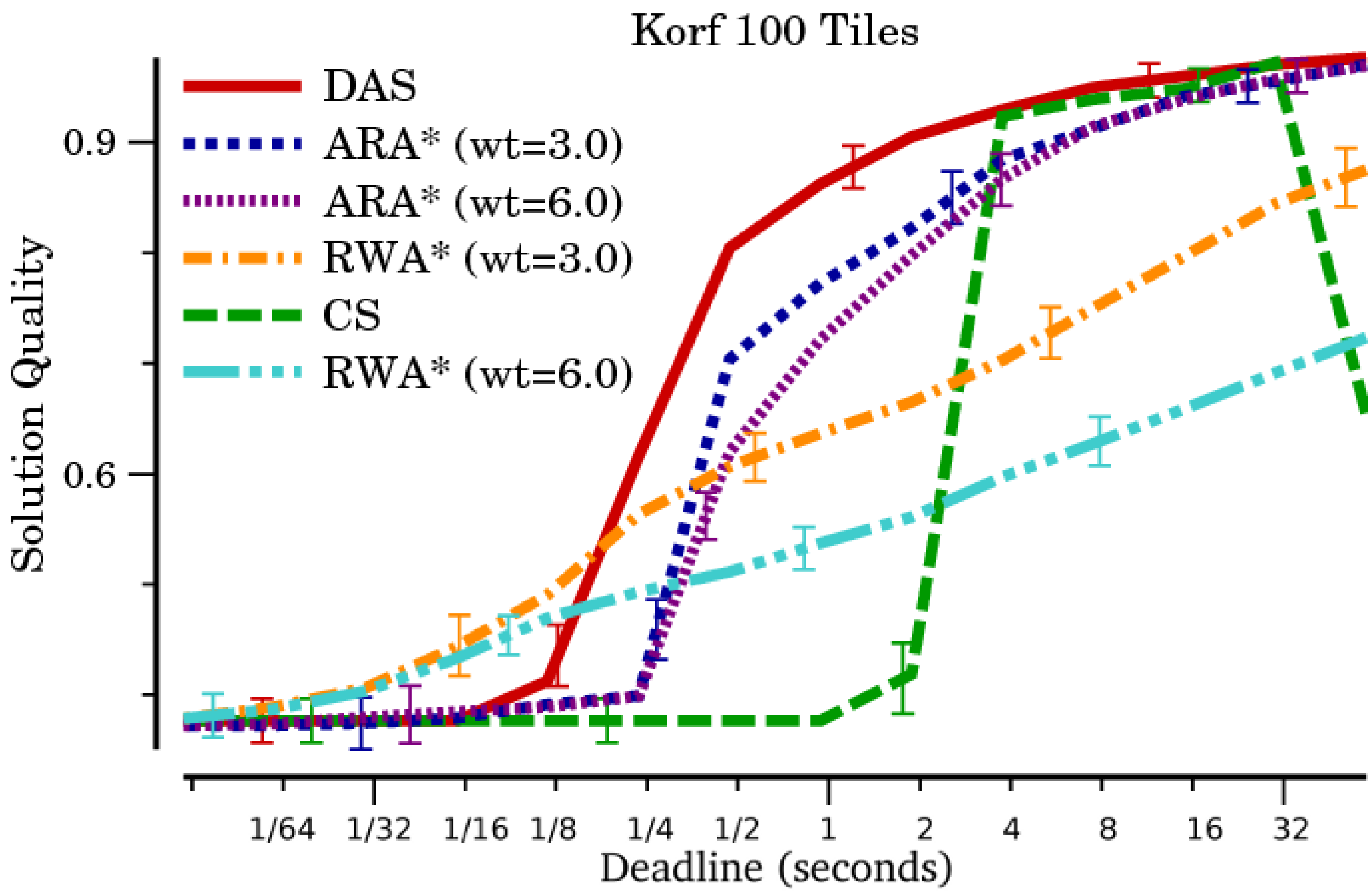
source #3 of 5: search vacillation

$$d_{max} = \text{time remaining} \,/\, \overline{\Delta e}$$

Deadline-Aware Search (Dionne et al, SoCS-11)

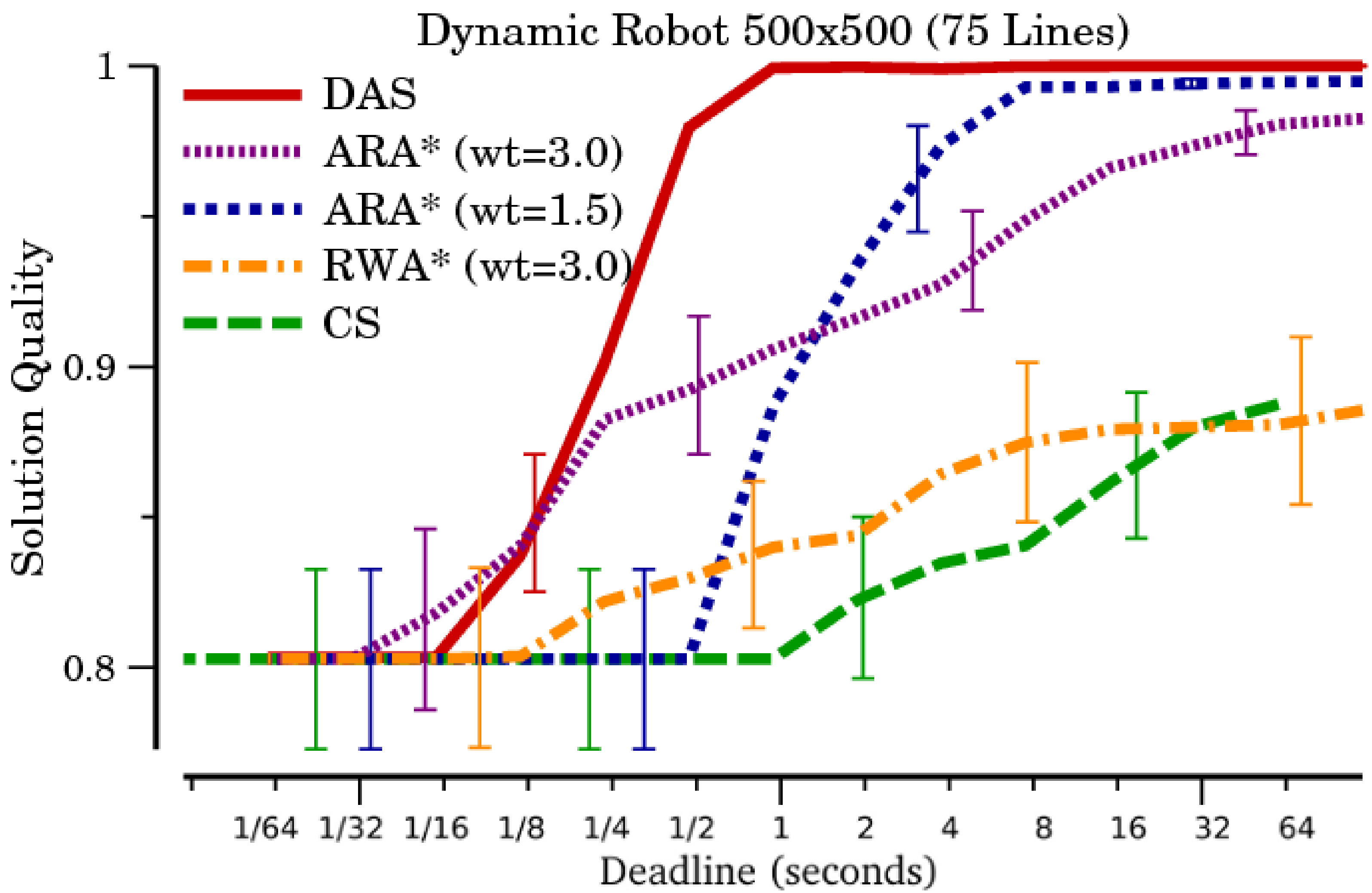
while time remains and *open* is not empty
    $d_{max} \leftarrow$ estimate bound
    $s \leftarrow$ pop $best_f$ from *open*
    if s is a goal
        save if best so far
    else if $\widehat{d}(s) < d_{max}$
        expand s
    else
        add s to *pruned*
    if *open* is empty, recover using *pruned*

DAS Results



Korf 100 Tiles
DAS
ARA* (wt=3.0)
ARA* (wt=6.0)
RWA* (wt=3.0)
CS
RWA* (wt=6.0)
Solution Quality
0.9
0.6
1/64
1/32
1/16
1/8
1/4
1/2
1
2
4
8
16
32
Deadline (seconds)

DAS Results



Dynamic Robot 500x500 (75 Lines)
DAS
ARA* (wt=3.0)
ARA* (wt=1.5)
RWA* (wt=3.0)
CS
Solution Quality
1
0.9
0.8
1/64
1/32
1/16
1/8
1/4
1/2
1
2
4
8
16
32
64
Deadline (seconds)

Search with a Utility Function

Utility-based Search

balance time and cost according to user's utility function

Example: linear utility function

for solution of cost f produced after time t:

$$U(f,t) = -w_f \cdot f - w_t \cdot t$$

Utility-based Search

balance time and cost according to user's utility function

Example: linear utility function

for solution of cost f produced after time t:

$$U(f,t) = -w_f \cdot f - w_t \cdot t$$

Example: minimize goal achievement time

if cost = plan makespan,

$$U(f,t) = -f - t$$

balance time and cost according to user's utility function

Example: linear utility function

for solution of cost f produced after time t:

$$U(f,t) = -w_f \cdot f - w_t \cdot t$$

Example: minimize goal achievement time

if cost = plan makespan,

$$U(f,t) = -f - t$$

anytime algorithms?

Problems with Anytime Algorithms

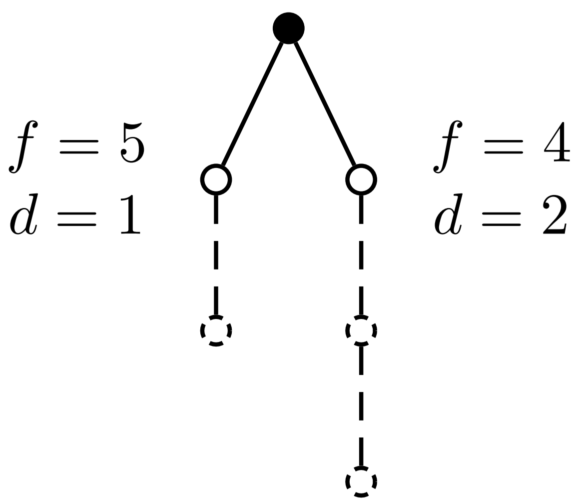
Requires a termination policy, assuming:

1. relevant solver features for predicting progress are known
2. training data available
3. new instance is similar in relevant aspects to training
4. relevant instance aspects are known

Requires a termination policy, assuming:

1. relevant solver features for predicting progress are known
2. training data available
3. new instance is similar in relevant aspects to training
4. relevant instance aspects are known

Impossible to design optimally:



Must know the user's trade-off!

Best-first Utility-guided Search, Yes! (BUGSY, IJCAI-07)

source #4 of 5: user's true objective/utility function

best-first search according to utility:

$$U(n) = \max_{s \text{ under } n} (-w_f \cdot f(s) - w_t \cdot t(s))$$

Best-first Utility-guided Search, Yes! (BUGSY, IJCAI-07)

source #4 of 5: user's true objective/utility function

best-first search according to utility:

$$U(n) = \max_{s \text{ under } n} (-w_f \cdot f(s) - w_t \cdot t(s))$$

convert $\widehat{d}(n)$ to $t(n)$

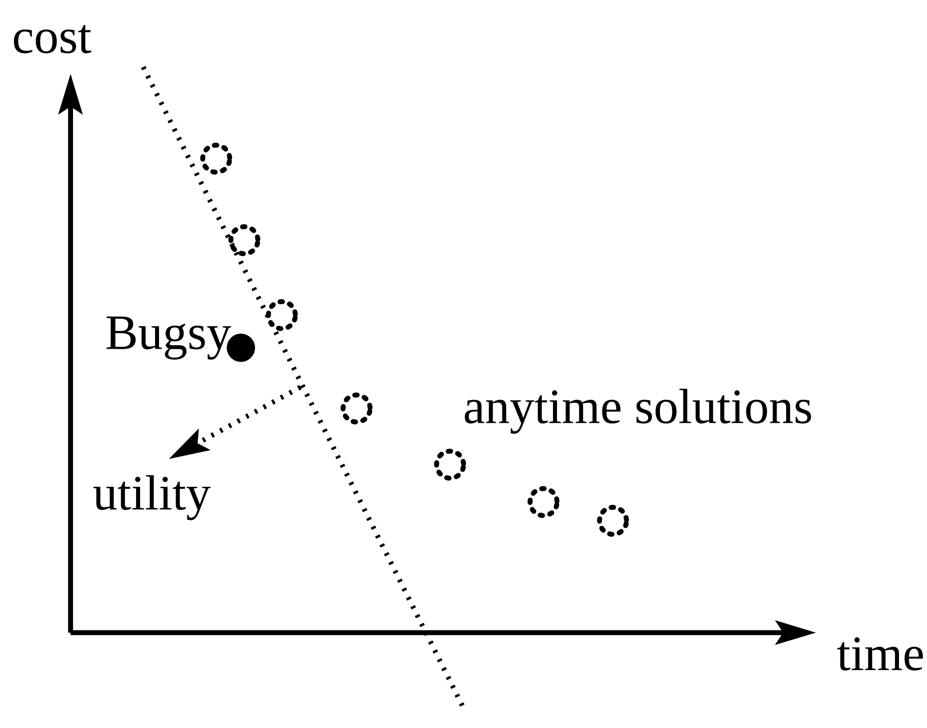
approximate s under n by cheapest and nearest

- need $d_{cheapest}$, $d_{nearest}$, $h_{cheapest}$, $h_{nearest}$
- source #5 of 5: cost-to-go to nearest, $h_{nearest}$
- seems straightforward in many domains

Properties

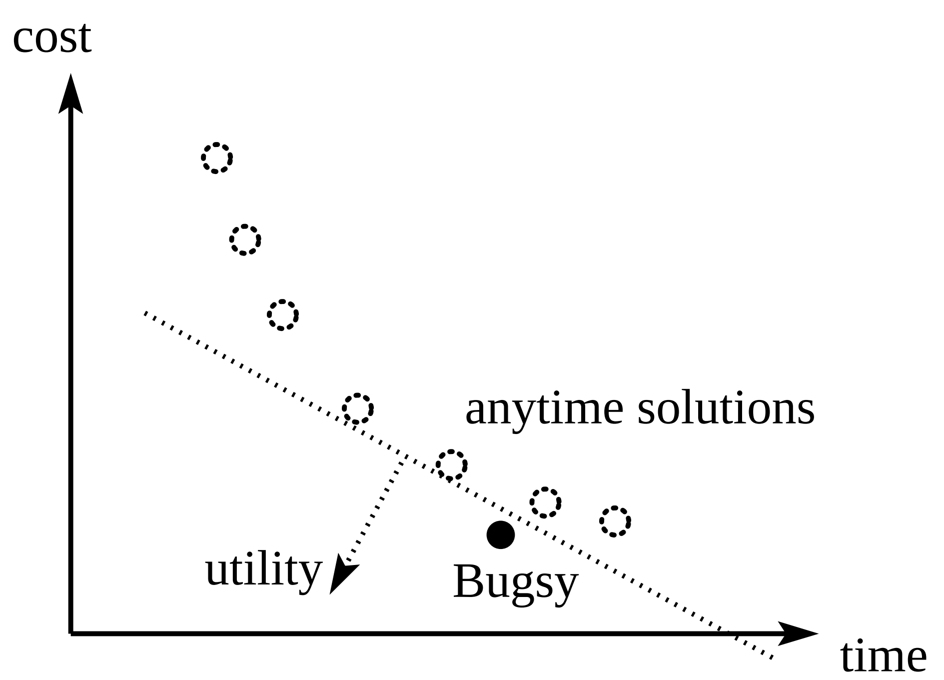
Different from anytime algorithms

- no need for termination policy (training data, precomputation)
- can spend all effort pursuing one solution
- no fixed trade-off

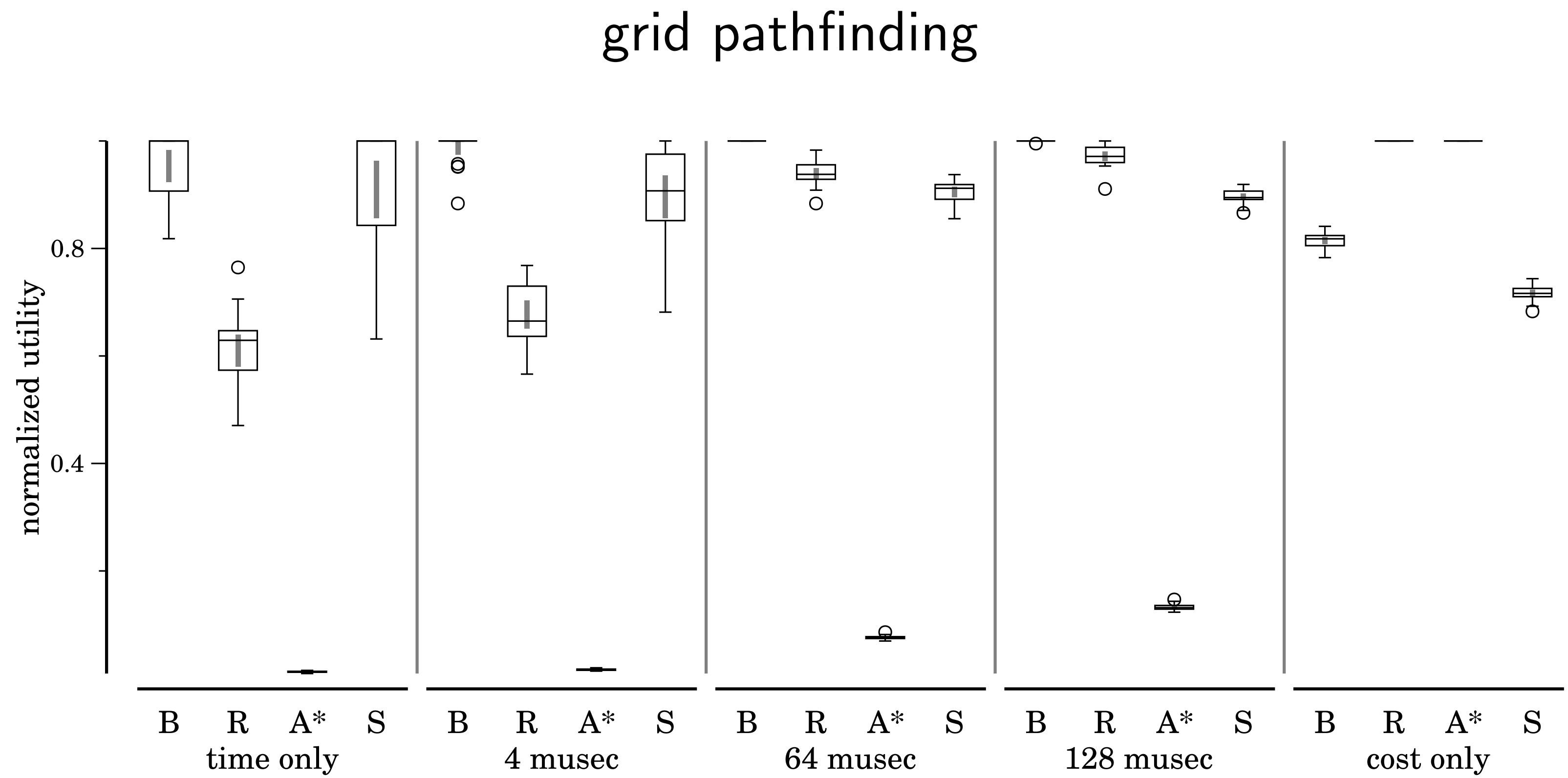


Different from anytime algorithms

- no need for termination policy (training data, precomputation)
- can spend all effort pursuing one solution
- no fixed trade-off



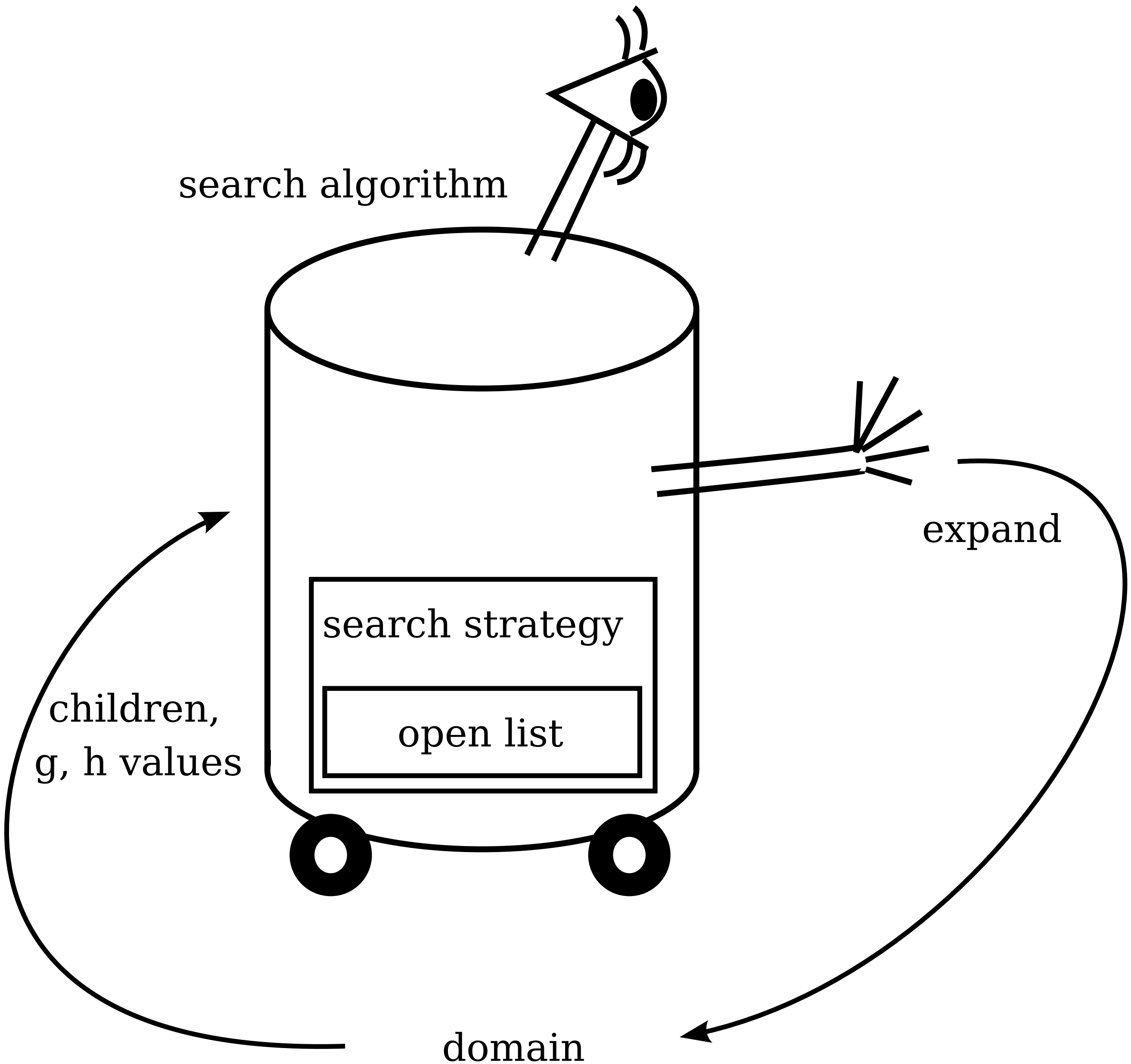
BUGSY Results



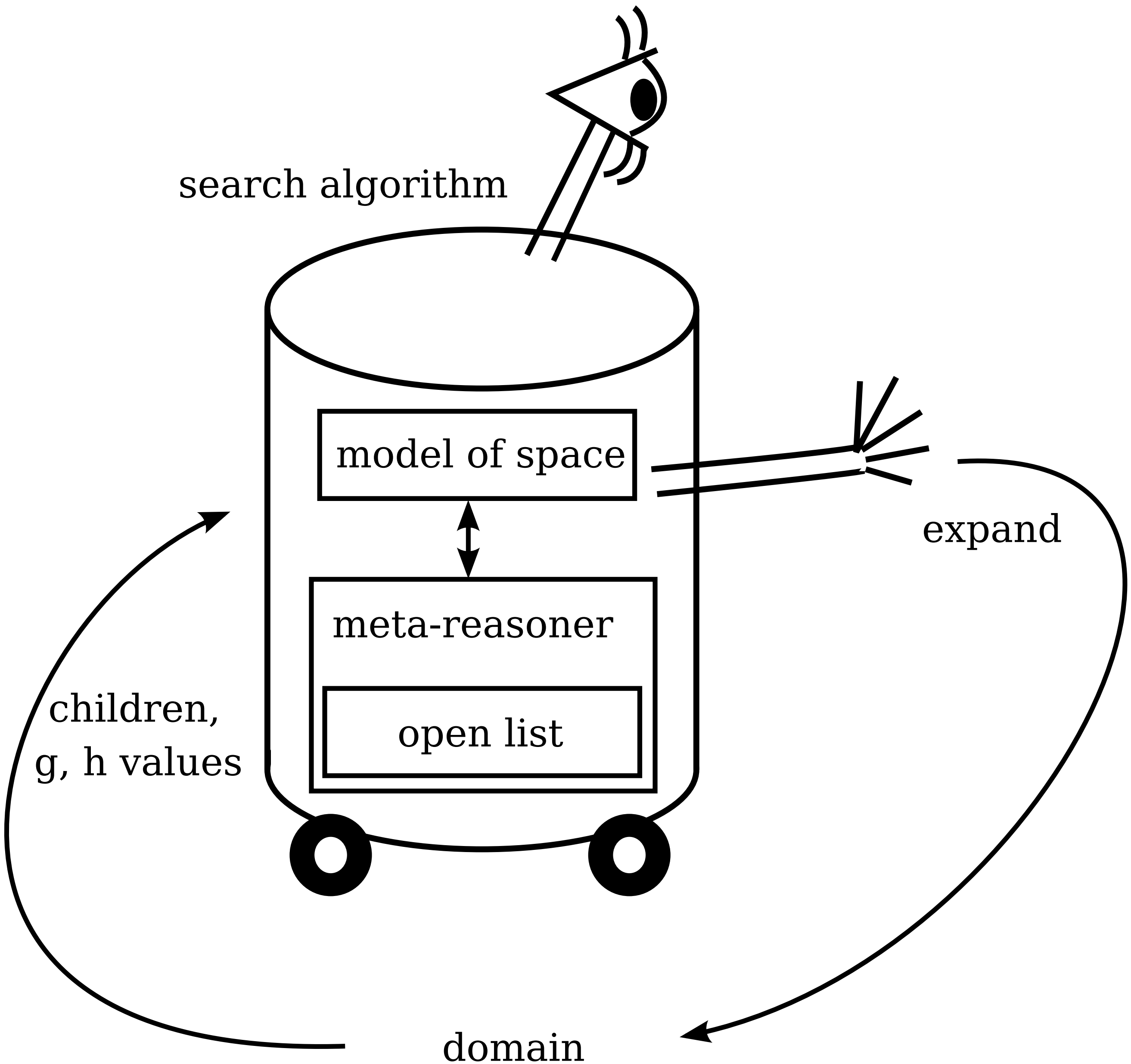
B — BUGSY
R — ARA* with termination policy learned off-line
A* — A*
S — Speedy

Conclusion

Rational Search



Rational Search



The Search Problem

Traditional:

1. initial state
2. goal predicate
3. expand — yields $g(n)$
4. $h(n)$

The Search Problem

Traditional:

1. initial state
2. goal predicate
3. expand — yields $g(n)$
4. $h(n)$

Possible:

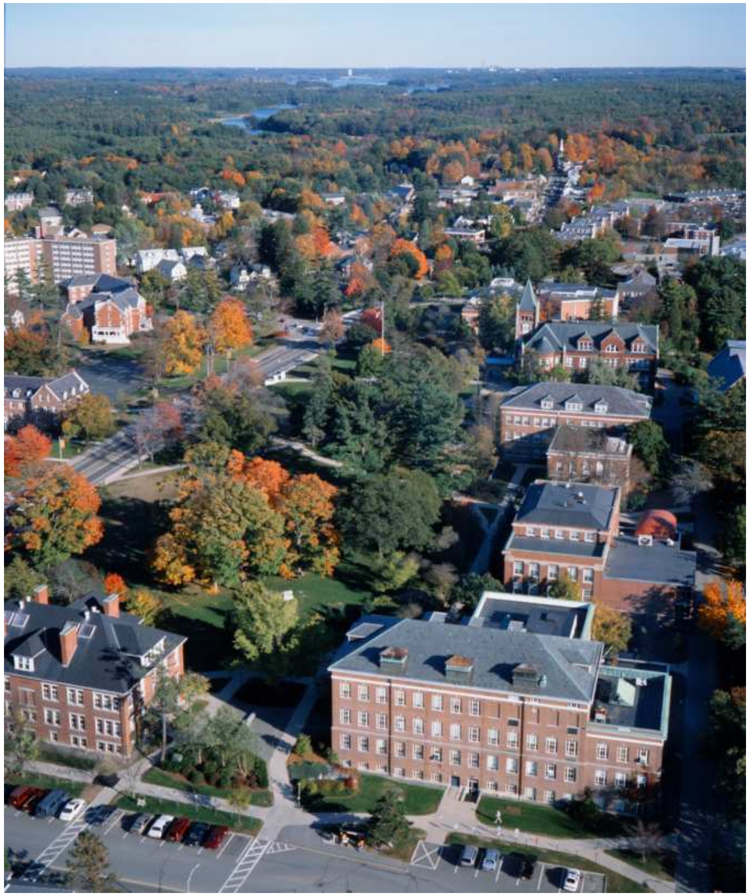
5. $d(n)$: distance-to-go
6. $\widehat{f}(n)$: expected cost, inadmissible h (expansion experience)
7. $\overline{\Delta e}$: expansion delay, time to goal (expansion experience)
8. $U(f,t)$: user's utility function, true objective
9. $h_{nearest}(n)$: cost-to-go to nearest

Summary

- search algorithms as agents
  - expansion as sensor reading instead of proof step!
  - connections:
    - reinforcement learning
    - metareasoning
    - decision-making
- more information sources

  what else can we exploit? and how?
- more problem settings
  - greedy search: Speedy
  - bounded-suboptimal search: EES
  - contract search: DAS
  - utility-based search: BUGSY

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