Suboptimal Heuristic Search

Wheeler Ruml

University of New Hampshire

All actual work done by my fantastic students and collaborators.
Grateful thanks to NSF, BSF, and DARPA.

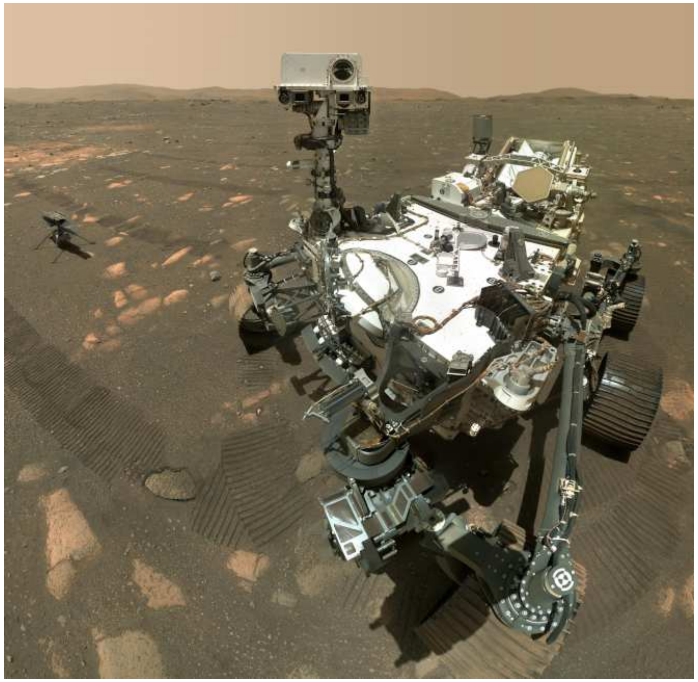
Heuristic Search Is Fundamental

Search enables planning / action selection

- achieve goal robustly
- optimize resource use (time, energy, pollution, . . .)
- autonomy or decision-support
- support retaskability, AGI

Search enables algorithms

- dynamic programming
- discrete optimization
- ‘intractable’ $\rightarrow$ possible



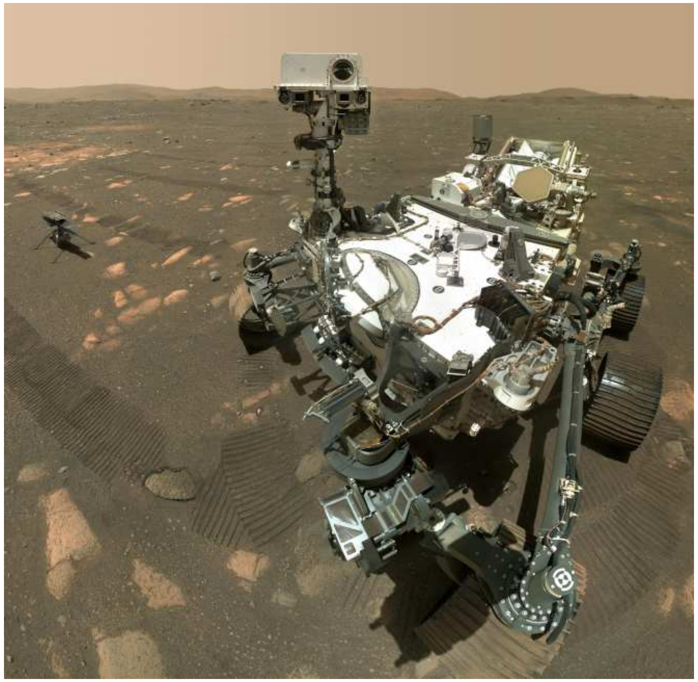
Heuristic Search Is Fundamental

Search enables planning / action selection

- achieve goal robustly
- optimize resource use (time, energy, pollution, . . .)
- autonomy or decision-support
- support retaskability, AGI

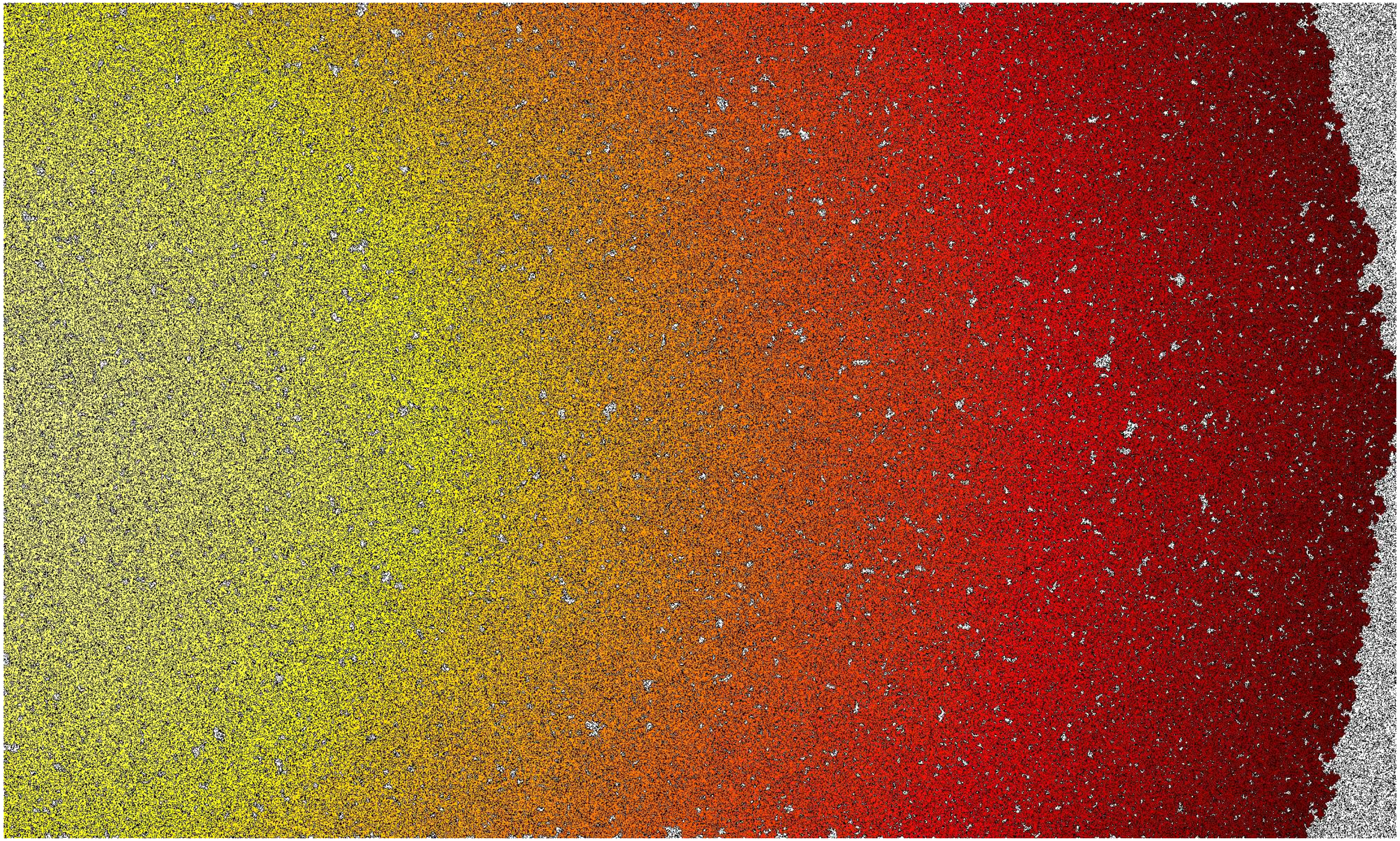
Search enables algorithms

- dynamic programming
- discrete optimization
- 'intractable' $\rightarrow$ possible



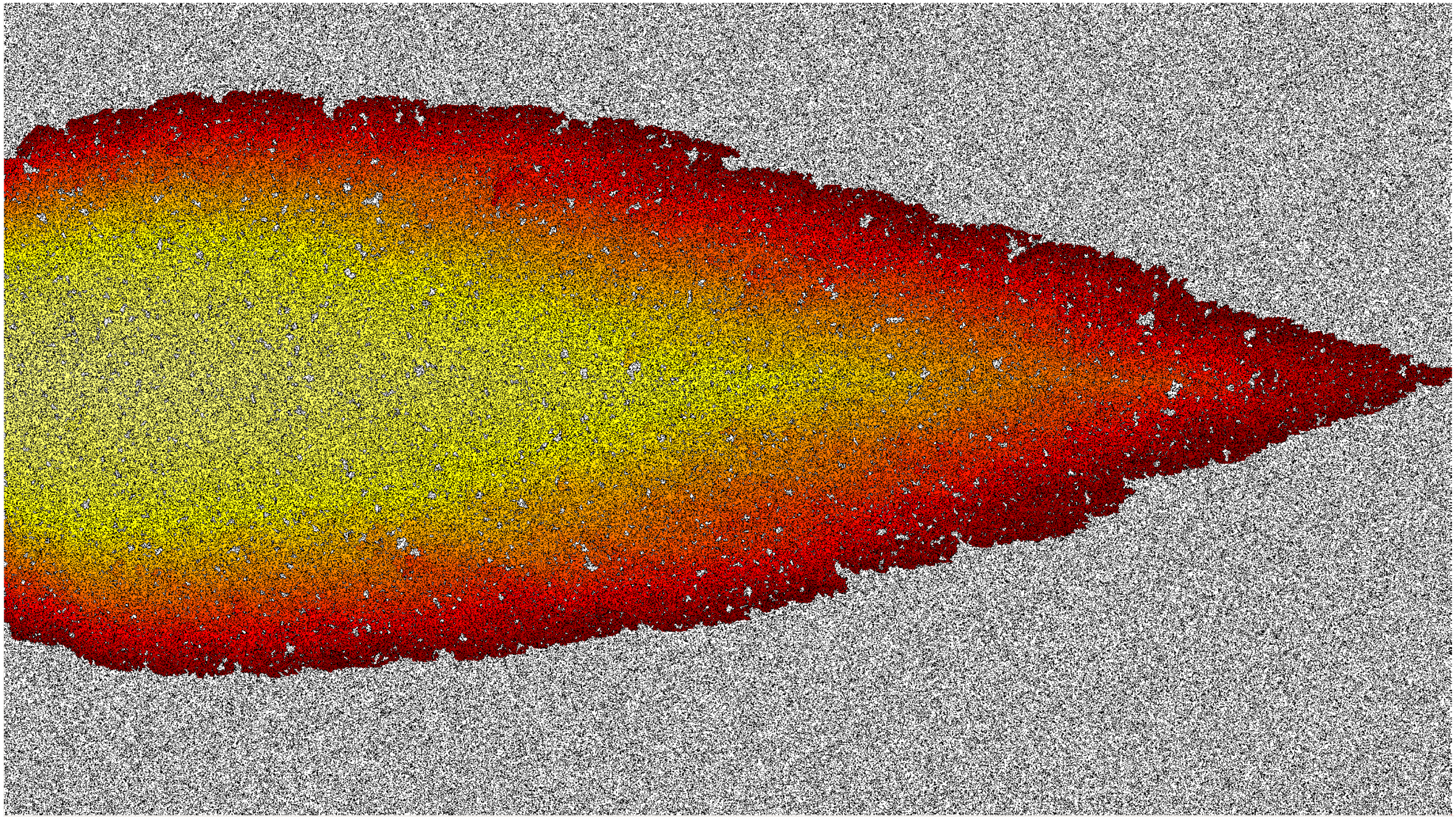
Point 1/3: Suboptimal search is the most important kind!

Dijkstra vs A* for Pathfinding



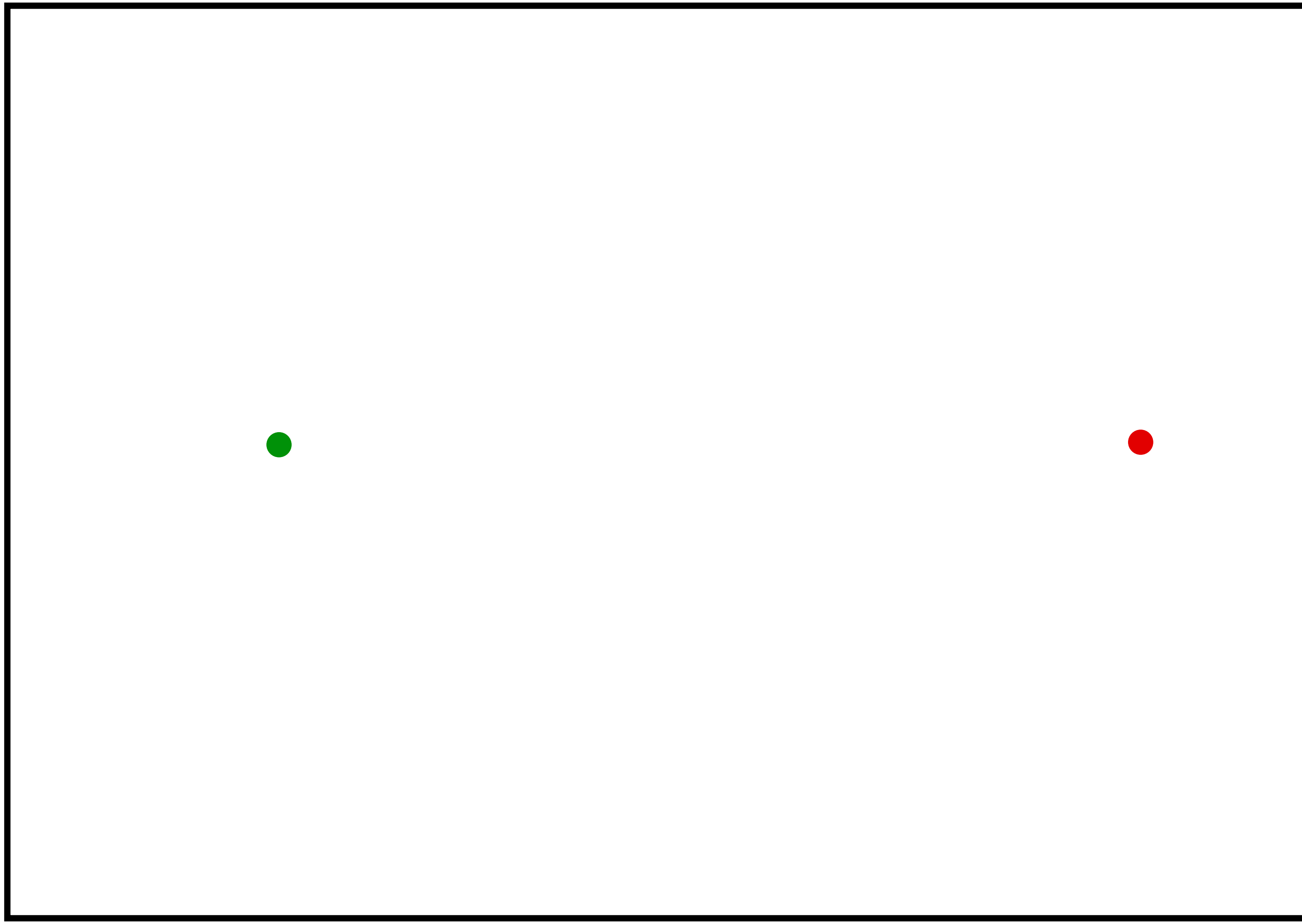
uniform-cost search: best-first on g

Dijkstra vs A* for Pathfinding

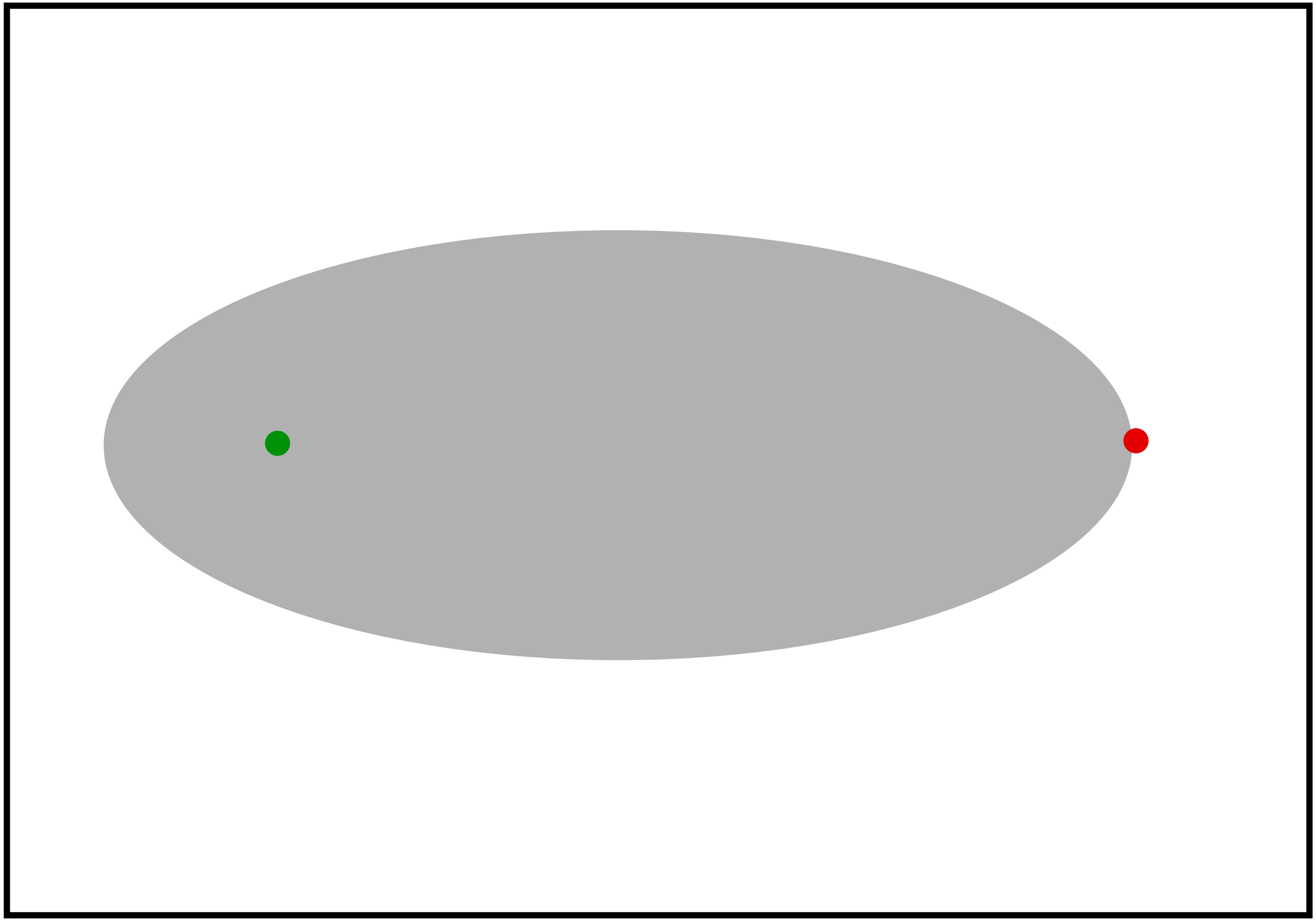


A*: best-first on $f = g + h$

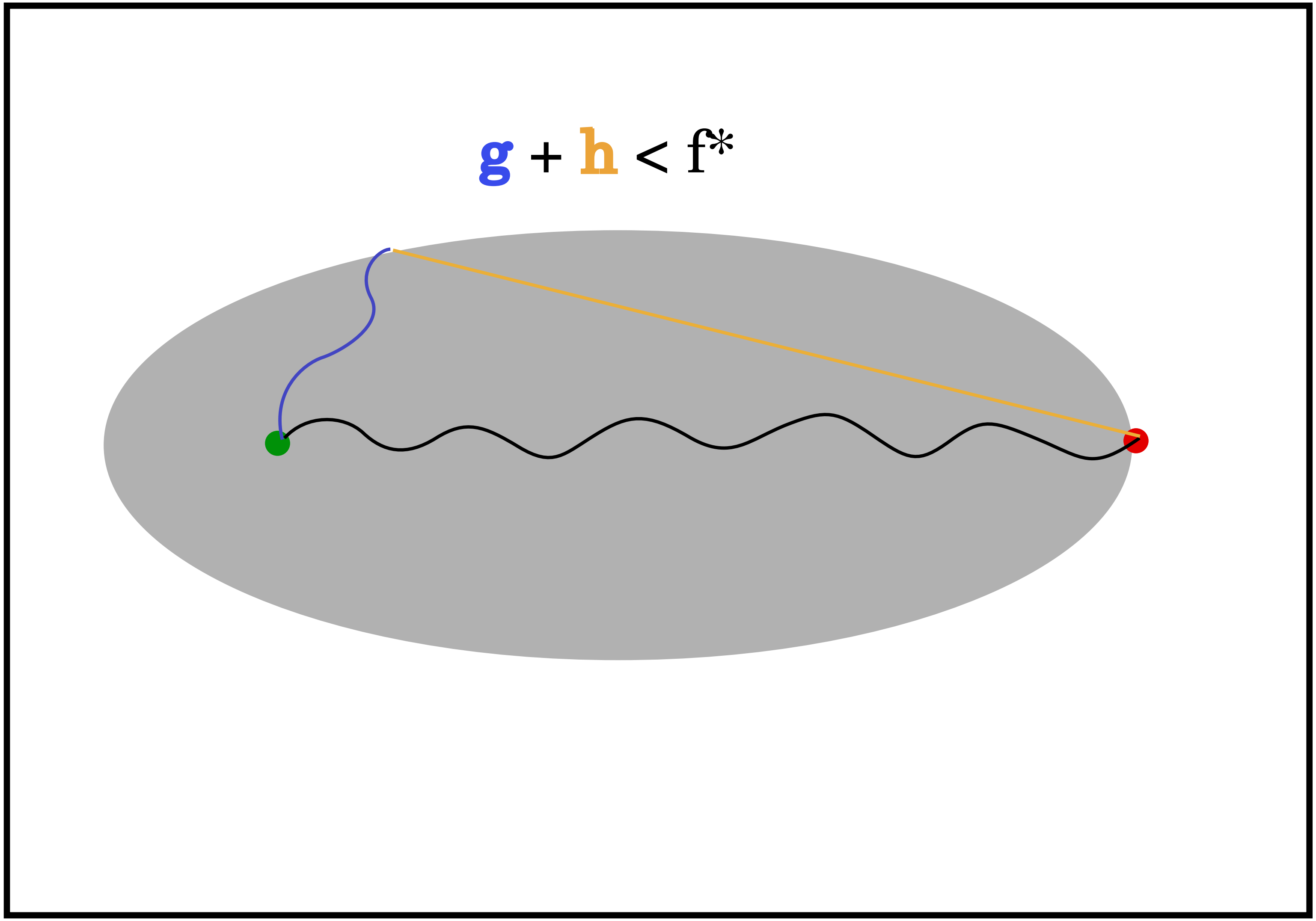
A* (Hart, Nilsson, and Raphael, 1968)



A* (Hart, Nilsson, and Raphael, 1968)



A* (Hart, Nilsson, and Raphael, 1968)



heuristic is more about procrastination or pruning than guidance

Optimal Search is Provably Impractical

all nodes with $g(n) + h(n) < f^*$

Helmert and Röger, "How Good is Almost Perfect?", AAAI-08 Best Paper Award:

> In many cases, such as the GRIPPER domain and a family of MICONIC tasks, there is no significant difference in node expansions between A* with an almost perfect heuristic and breadth-first search.

> We suggest that, beyond a certain point, trying to improve a heuristic search algorithm by refining its heuristic estimates is basically fruitless.

Optimal Search is Provably Impractical

all nodes with $g(n) + h(n) < f^*$

Helmert and Röger, "How Good is Almost Perfect?", AAAI-08 Best Paper Award:

> In many cases, such as the GRIPPER domain and a family of MICONIC tasks, there is no significant difference in node expansions between A* with an almost perfect heuristic and breadth-first search.

> We suggest that, beyond a certain point, trying to improve a heuristic search algorithm by refining its heuristic estimates is basically fruitless.

detecting symmetries and partial orders only fixes modeling errors

suboptimal search is the practical answer!

Suboptimal Search Problem Settings

optimal: minimize solution cost

Suboptimal Search Problem Settings

optimal: minimize solution cost

---

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

tunable: trade cost for time, no guarantees

anytime: incrementally converge to optimal

situated: minimize goal achievement time

utility: maximize function of cost and time

real-time: return next action within absolute time bound

Suboptimal Search Problem Settings

optimal: minimize solution cost

---

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

tunable: trade cost for time, no guarantees

anytime: incrementally converge to optimal

situated: minimize goal achievement time

utility: maximize function of cost and time

real-time: return next action within absolute time bound

My personal very biased view!

Taking Suboptimal Search Seriously: More Information

information that becomes available during problem-solving

Point 2/3: many sources of information beyond h!

$\widehat{h}(n)$: unbiased heuristics (possibly learned on-line from h)

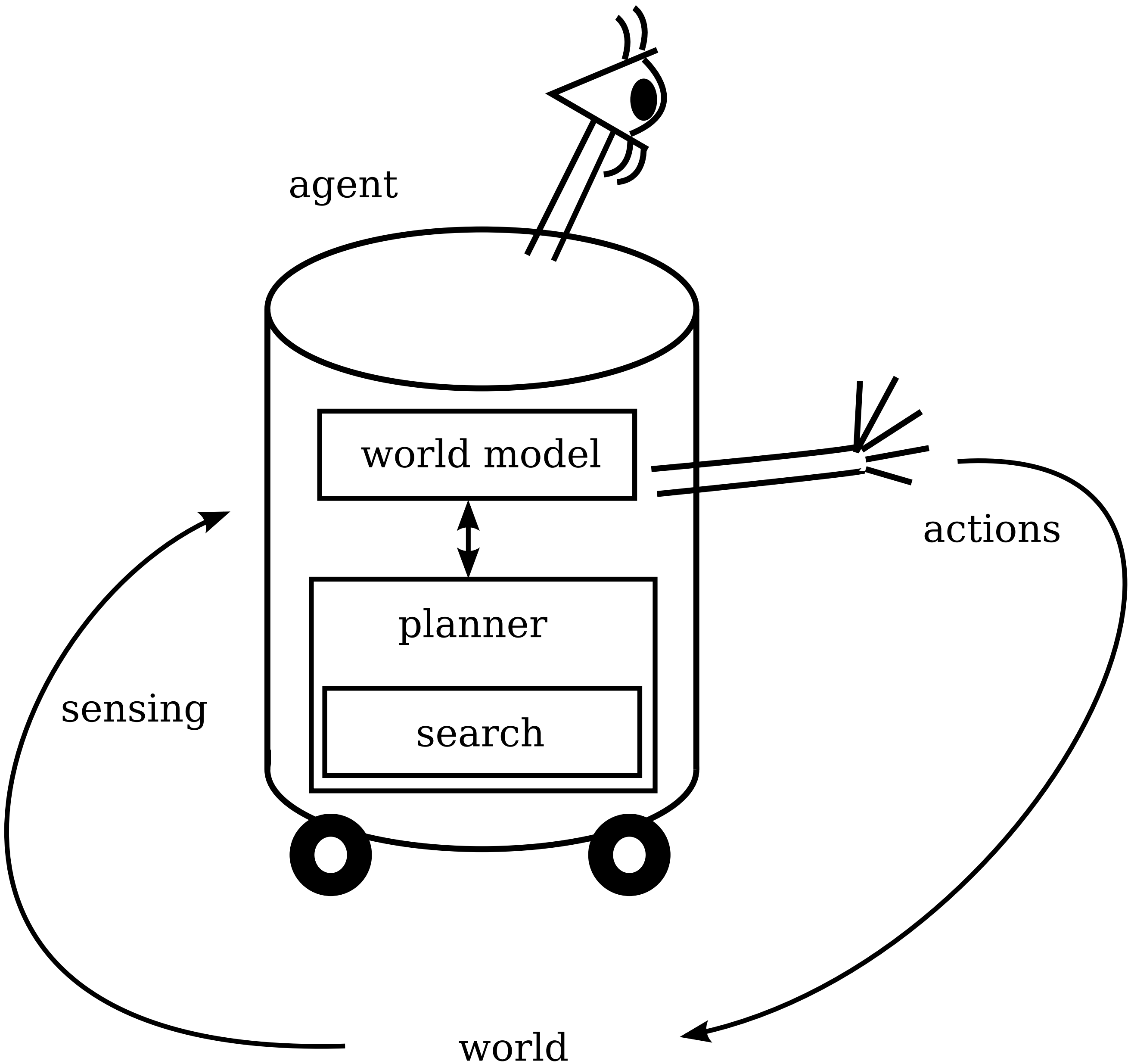
$\widehat{d}(n)$: distance-to-go estimates (eg, unit-cost h)

experience so far: eg, how misleading are estimates? how many paths look promising?

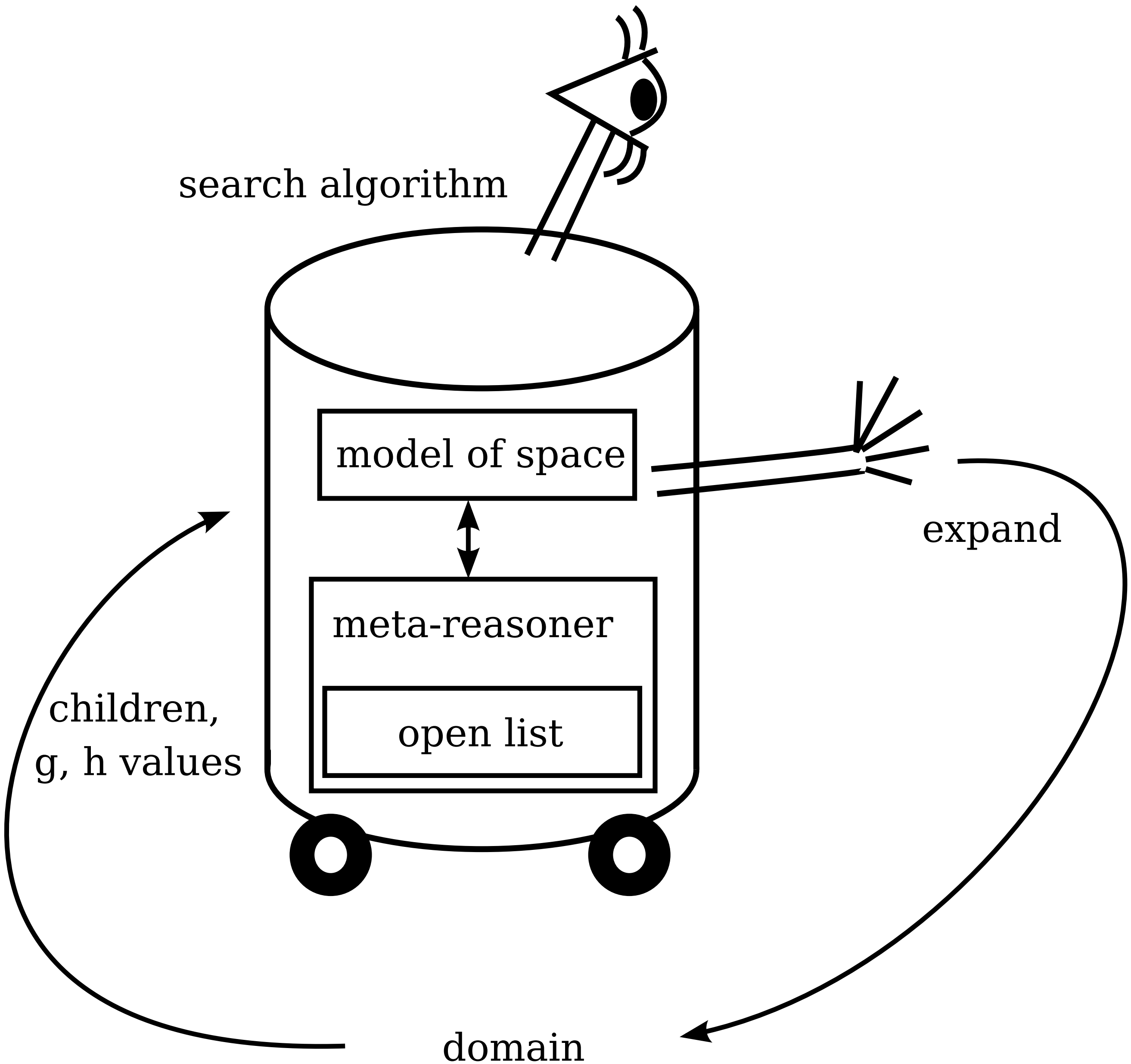
beliefs: distributions over values, quantify uncertainty

not today: preferred actions / policies

Classic AI Agent



Point 3/3: Search Algorithm as an Agent

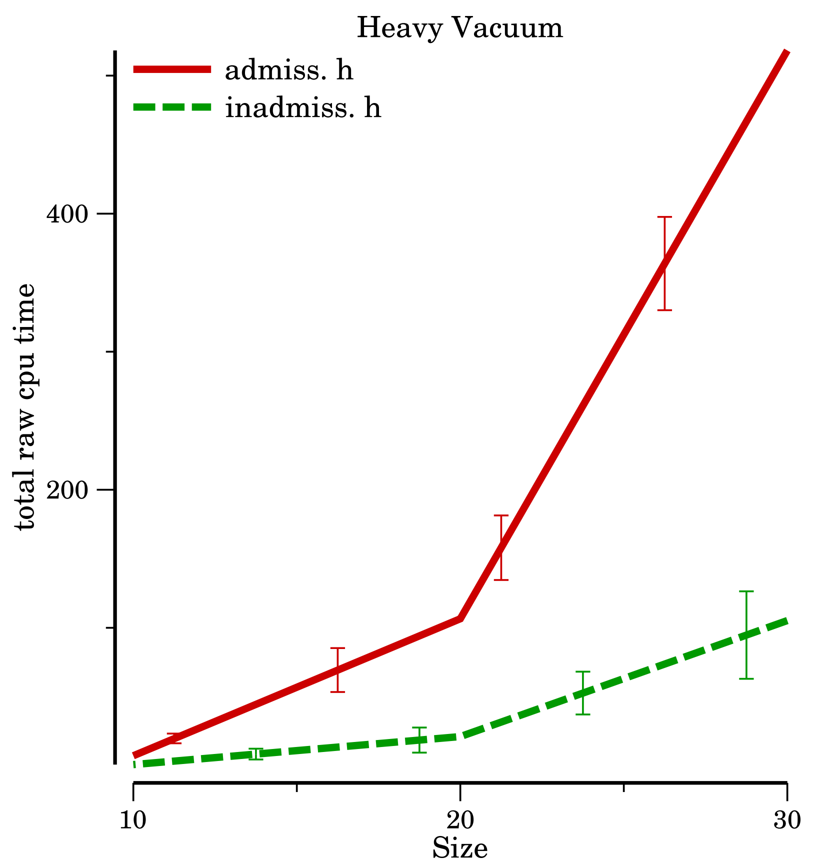


Greedy Search

Inadmissible Heuristics $\widehat{h}$

finding solutions as quickly as possible

Greedy best-first search (GBFS): best-first search on h
inadmissible $\widehat{h}$ can be more informed



searching on $\widehat{h}$ is faster than h

Distance-to-go

minimize solving time $=$ minimize number of expansions to goal

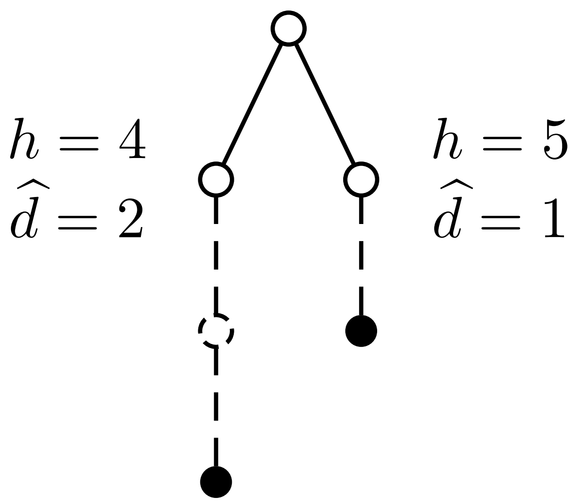
for domains with costs, this is not $h(n)$

Distance-to-go

minimize solving time = minimize number of expansions to goal

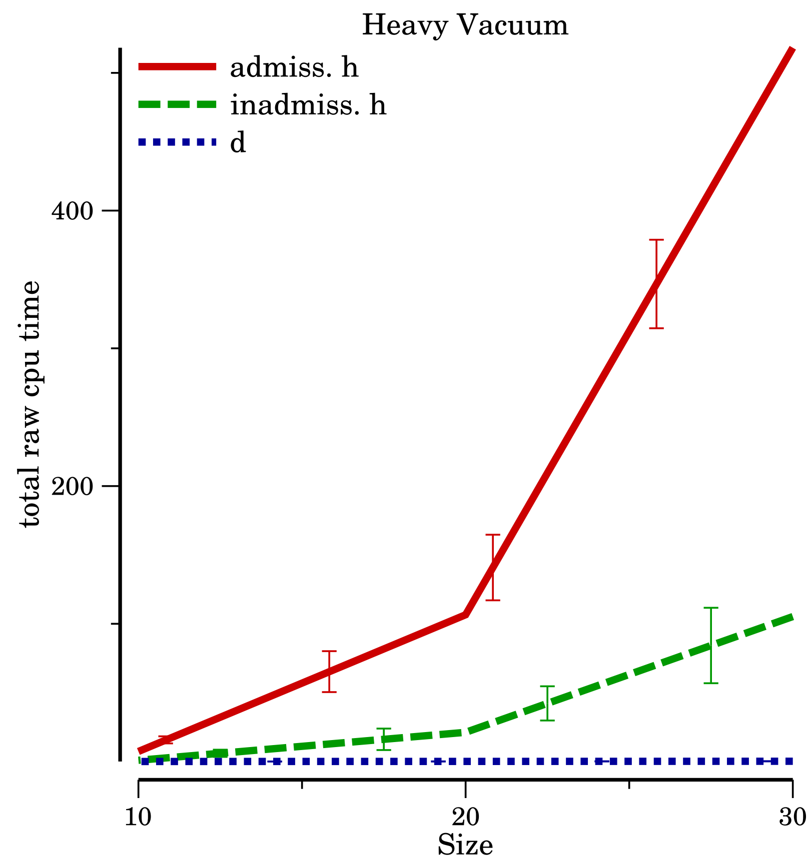
for domains with costs, this is not $h(n)$

$\widehat{d}(n)$ distance-to-go, remaining solution path length, arcs-to-go, hops-to-go



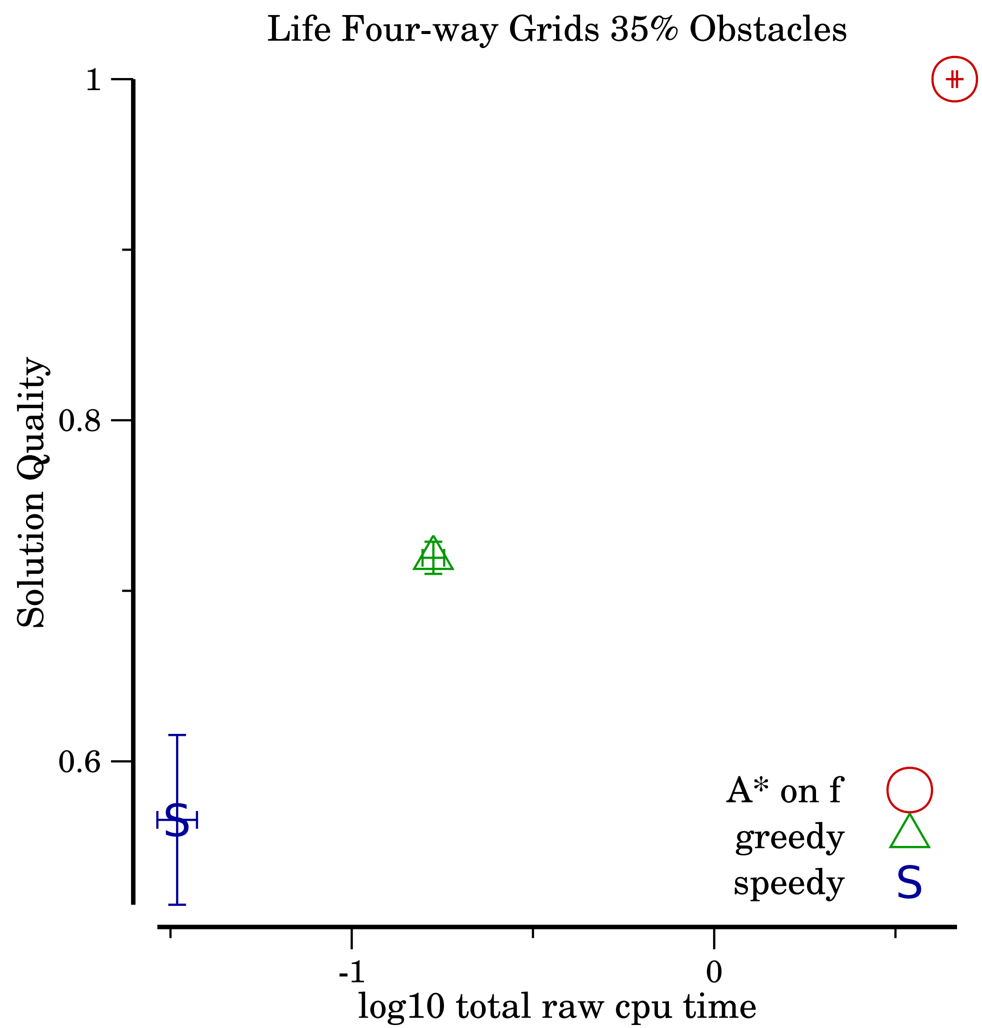
Performance of Distance-to-go

Greedy: best-first search on h
Speedy: best-first search on $\widehat{d}$ (Thayer, Ruml, and Kreis, SoCS-09)



searching on $\widehat{d}$ is faster than $\widehat{h}$

Performance of Speedy Search



Speedy cost often reasonable (sometimes better than Greedy!)

Why Is Speedy Faster Than Greedy?

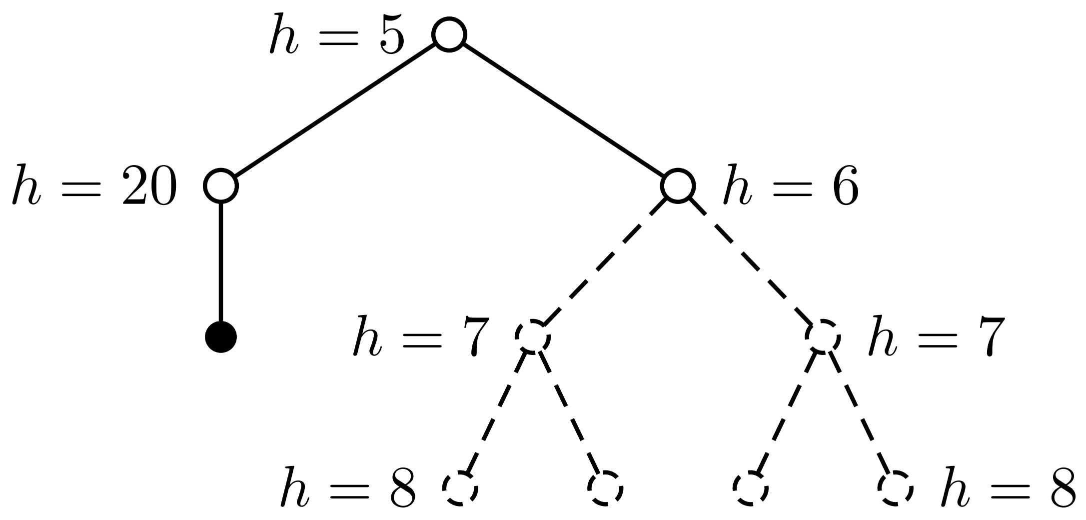
why faster than h? (Wilt and Ruml, SoCS-14)

- not: predicts search effort
- local minima are smaller

Domain	Cost	Max Local Min Size	Expected Min Size	Exp
Tiles	unit	392	2	801
	inverse	51,532	87	93,010
	rev inv	2091	2	855
Hanoi	unit	7,587	1,892	36,023
	rev sq	35,874	4,416	559,250
	square	2,034	201	4,663
TopSpin	unit	296	250	933
	sum	922	3	749
	stripe	240	3	441

Minimum Size Controls GBFS

intuition: for high cost ratios, many annoyingly cheap paths required to compensate for one unforeseen expensive action



Other Greedy Search Algorithms

robust greedy search is a wide open area!

exploration/diversity: ϵ-greedy, type-wA*

Bounded-suboptimal Search

quickly find a solution within factor b of optimal

simple hack (Pohl, AIJ 1970): $f'(n) = g(n) + b \cdot h(n)$

Bounded-suboptimal Search: Weighted A*

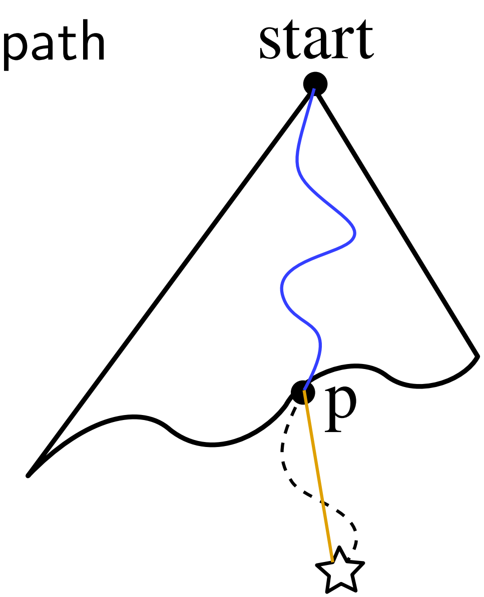
quickly find a solution within factor b of optimal

simple hack (Pohl, AIJ 1970): $f'(n) = g(n) + b \cdot h(n)$

f_{min} = lowest $f(n)$ on *open*

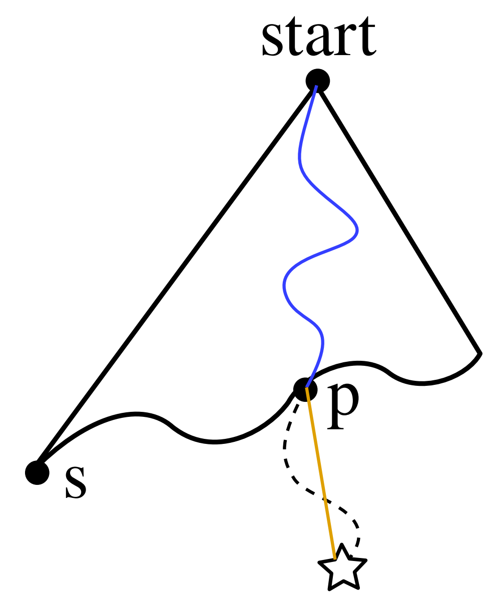
the key lemma: f_{min} is a global lower bound

1. any optimal path must pass through the frontier
2. let p be an open node along an optimal path
3. $f_{min} \leq f(p) = g(p) + h(p) \leq f^*$



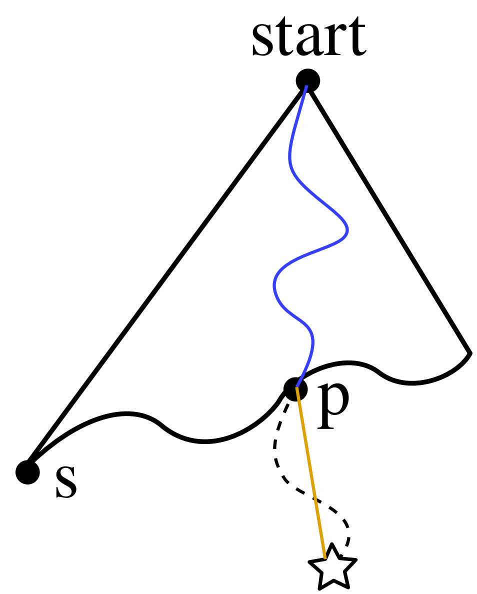
can expand any node with $f(n) \leq b \cdot f_{min}$!

Weighted A*'s Bounded Suboptimality



wA*'s bounded suboptimality:

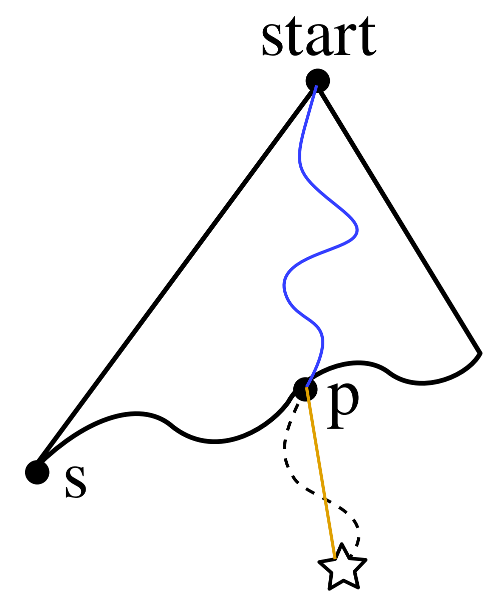
$$f'(s) \leq f'(p)$$



wA*'s bounded suboptimality:

$$g(s) = \quad f'(s) \leq f'(p) \quad = g(p) + b \cdot h(p)$$

Weighted A*'s Bounded Suboptimality

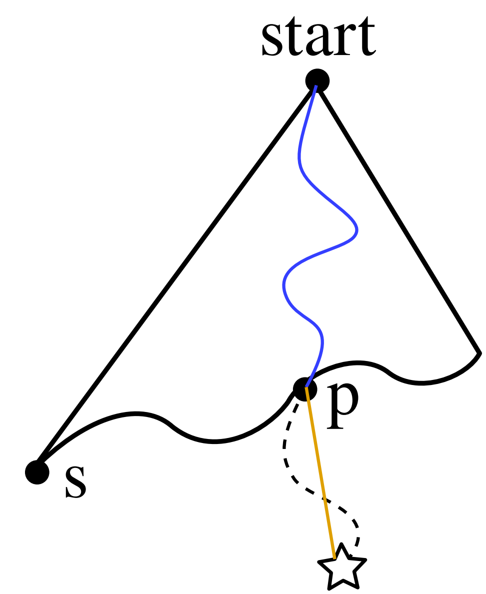


wA*'s bounded suboptimality:

$$g(s) = f'(s) \leq f'(p) = g(p) + b \cdot h(p)$$

$$\leq b \cdot f(p) \leq b \cdot f^*$$

Weighted A*'s Bounded Suboptimality



wA*'s bounded suboptimality:

$$g(s) = f'(s) \leq f'(p) = g(p) + b \cdot h(p) \leq b \cdot f(p) \leq b \cdot f^*$$

again: can expand any node with $f(n) \leq b \cdot f_{min}$!

Bounded-suboptimal Search: RR-d

quickly find a solution within factor b of optimal

two tasks: find sol $\leq b \cdot f_{min}$, raise f_{min}

RR-d (Fickert, Gu, and Ruml, AAAI-22):
multi-queue alternation (Röger and Helmert, ICAPS-10)

open: sorted on $\widehat{f}$, explore
focal: sorted on $\widehat{d}$, exploit
cleanup: sorted on f, raise bound

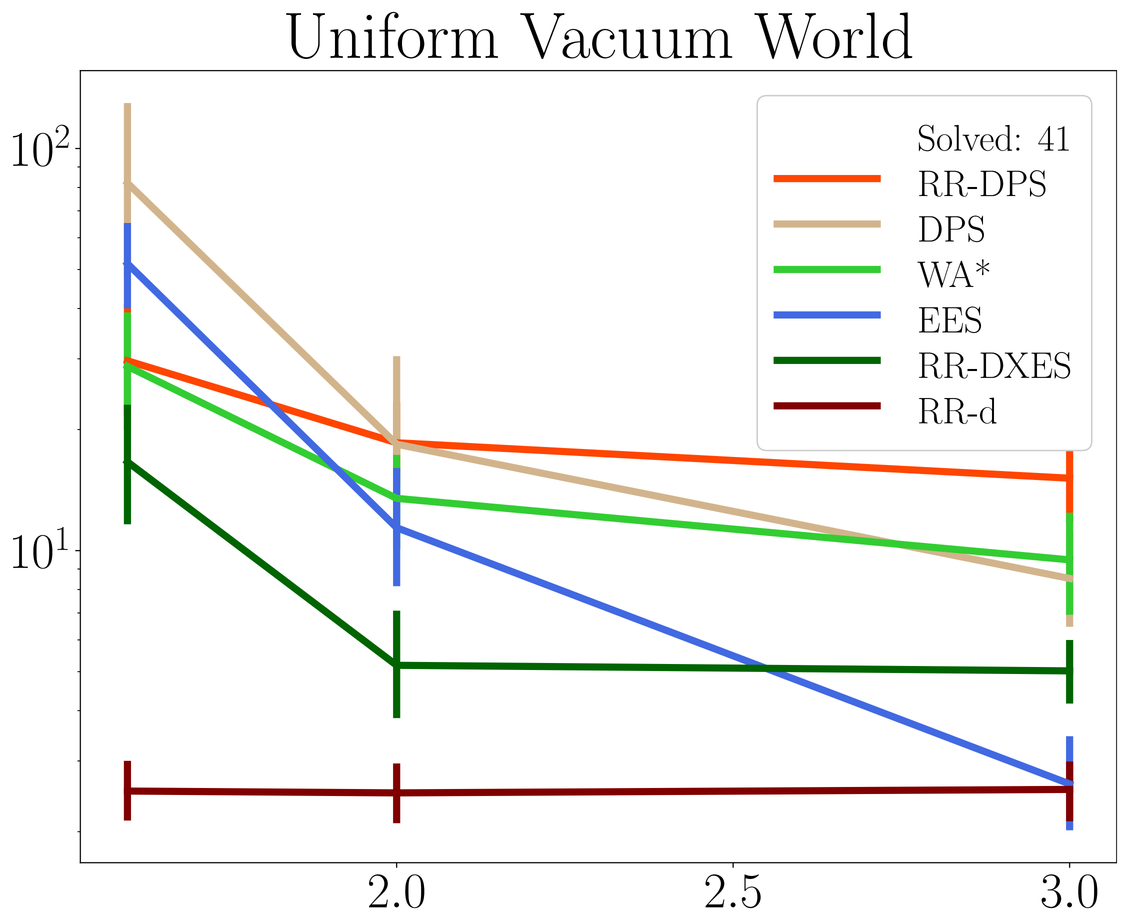
filter for open and focal: $f(n) \leq b \cdot f_{min}$

obvious ablations/substitutions are worse

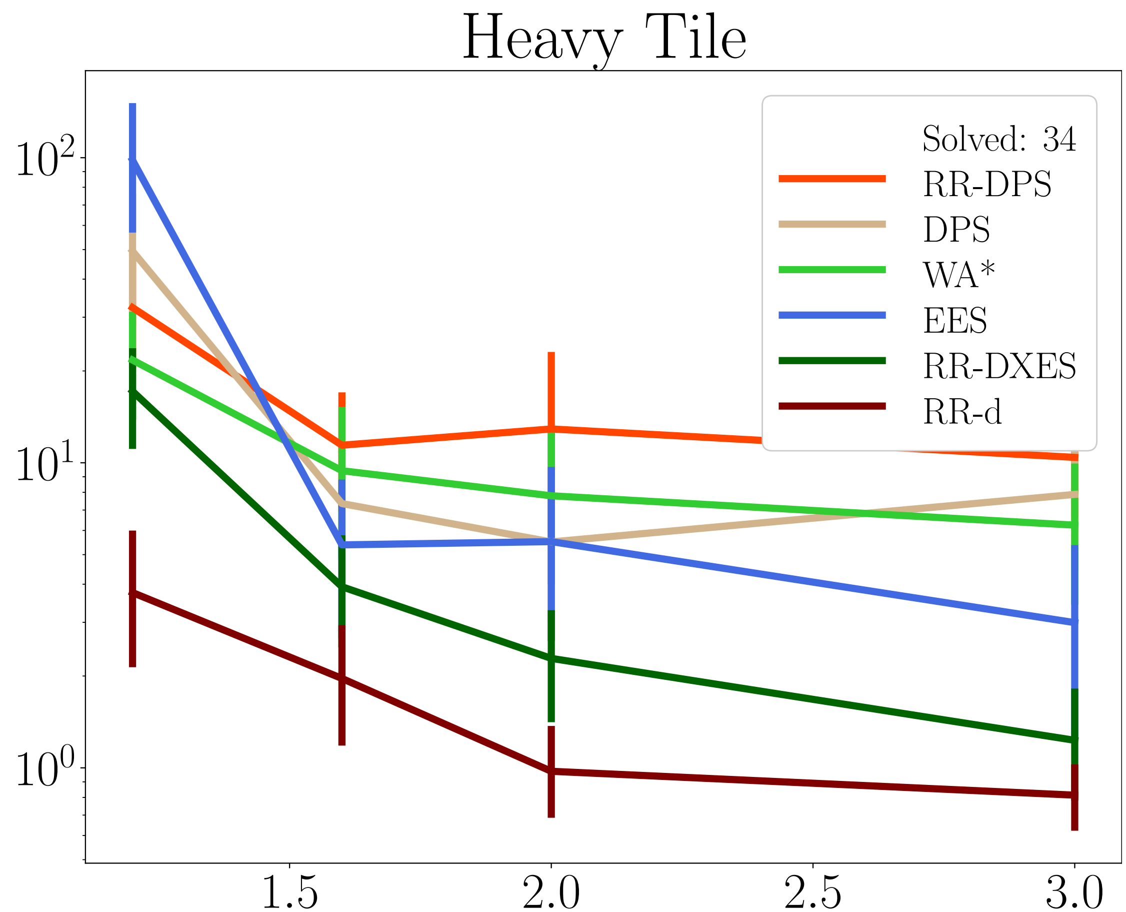
IPC Coverage ($b = 1.5$)

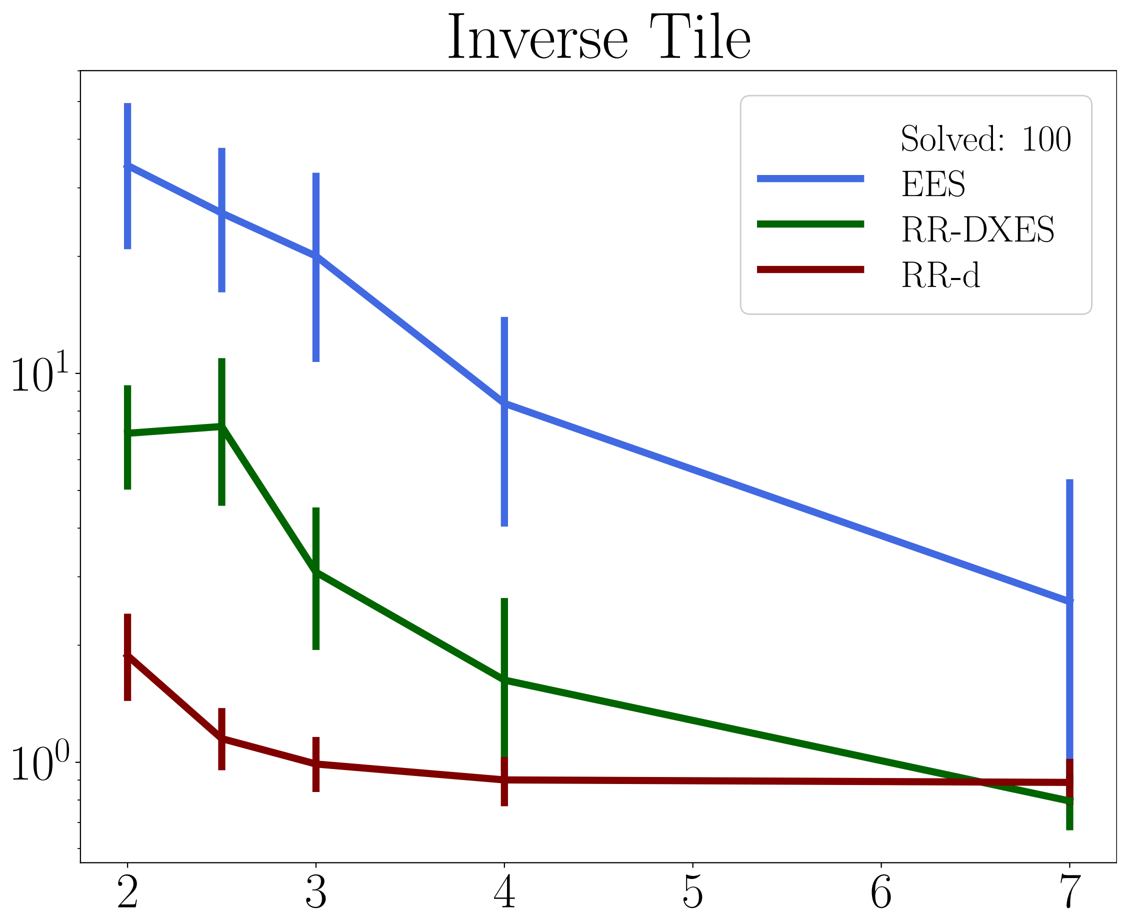
Coverage	WA*	EES	DPS	DXES	RR-DPS	RR-d
Sum (1652)	995	967	1012	894	982	**1025**
Normalized(%)	58.7	57.0	60.0	51.5	57.9	**60.7**
Expansions	569	558	472	734	665	**383**

Search Domains



Search Domains





many duplicates: Φ_{pwXD} (Chen and Sturtevant, AAAI-21)

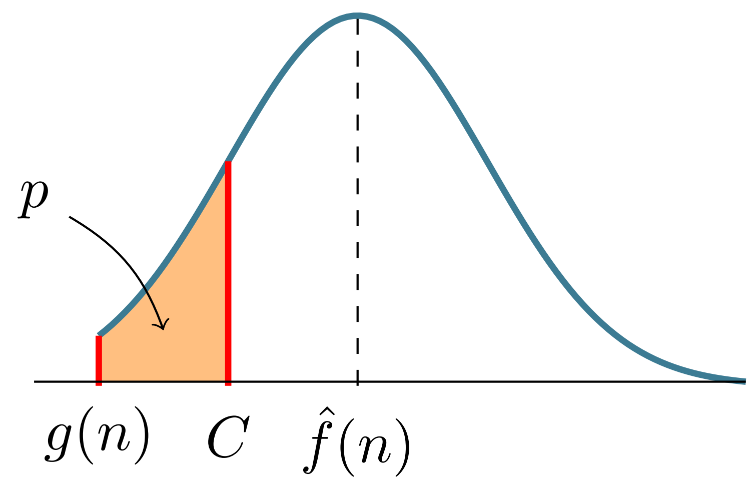
Bounded Cost Search

minimize time subject to absolute cost bound C

best-first on expected effort: $T(n)/p(n)$
(Dobson and Haslum, HSDIP-17)

$T(n)$: expected effort to find goal under n, eg $d(n)$

$p(n)$: probability of cost $< C$ under n
approximate from $g(n) + \hat{h}(n)$ belief distribution



XES on IPC'18 Bounded-Cost Track

Coverage	GBFS	PTS	BEES	BEES95	XES
Agricola (20)	**1**	0	0	0	0
Caldera (20)	8	10	10	12	**13**
Caldera-split (20)	**4**	2	2	2	2
DataNetwork (20)	2	0	3	3	**4**
Nurikabe (20)	4	10	10	**11**	9
Settlers (20)	4	5	10	**11**	**11**
Snake (20)	4	**5**	4	4	**5**
Spider (20)	7	**11**	10	10	9
Termes (20)	11	9	11	11	**13**
Sum (180)	45	52	60	64	**66**
Expansions $(*10^3)$	1.93	3.93	2.10	2.25	**1.77**

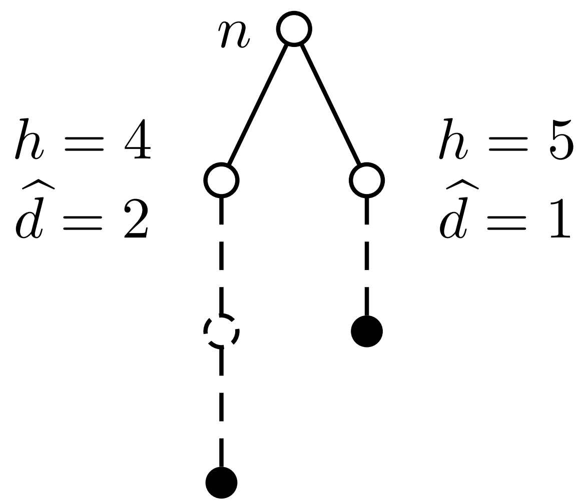
bounded-cost algs dominate GBFS, XES best overall

Contract Search

Ingredients for Contract Search

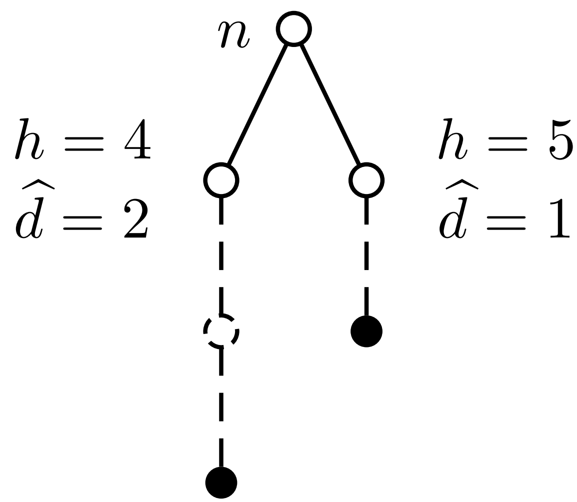
find cheapest solution within deadline

note: anytime algorithms (should) optimize for unknown deadline



Ingredients for Contract Search

find cheapest solution within deadline
note: anytime algorithms (should) optimize for unknown deadline



- $\widehat{d}(n)$ distance-to-go
- expansion delay: number of expansions when a node is generated and expanded
- $\widehat{d}(n) \cdot delay$ estimates expansions to goal
- time per expansion

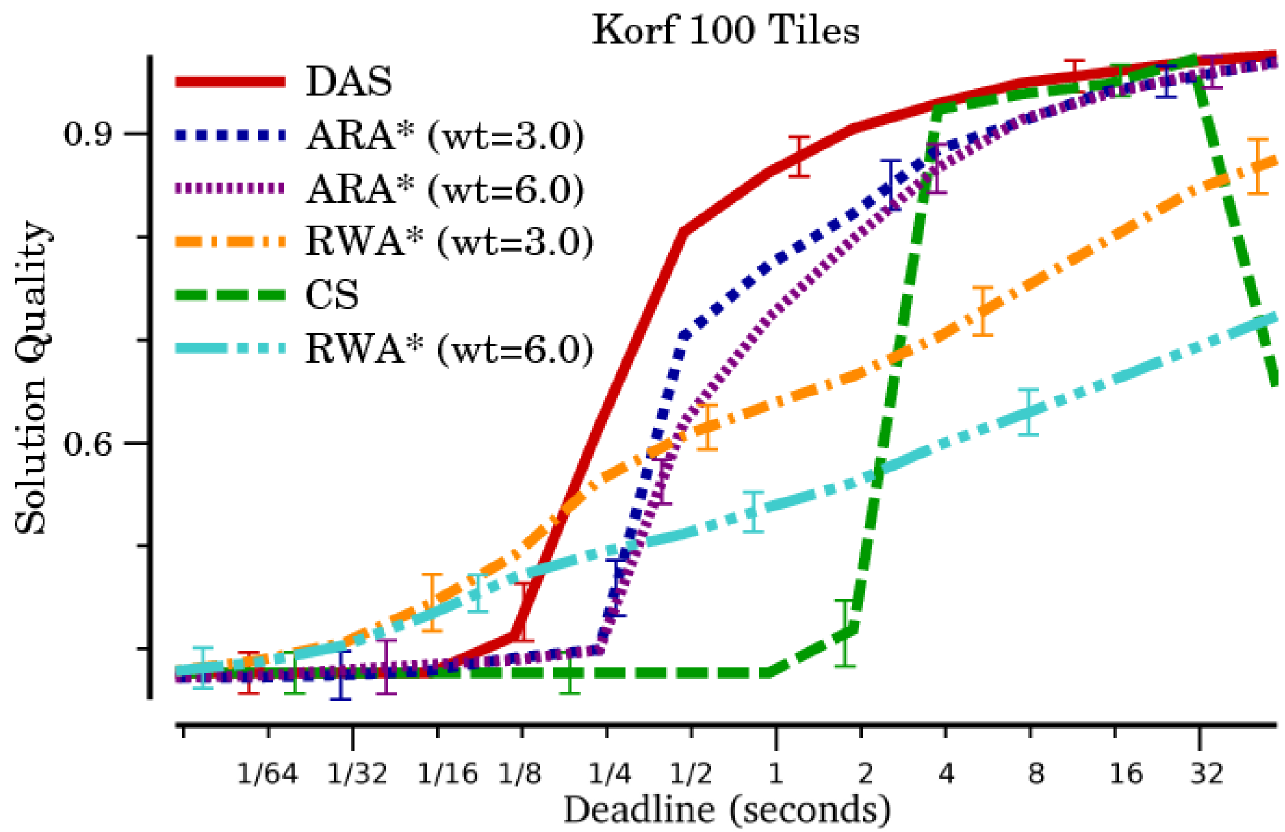
find cheapest solution within deadline

Deadline Aware Search

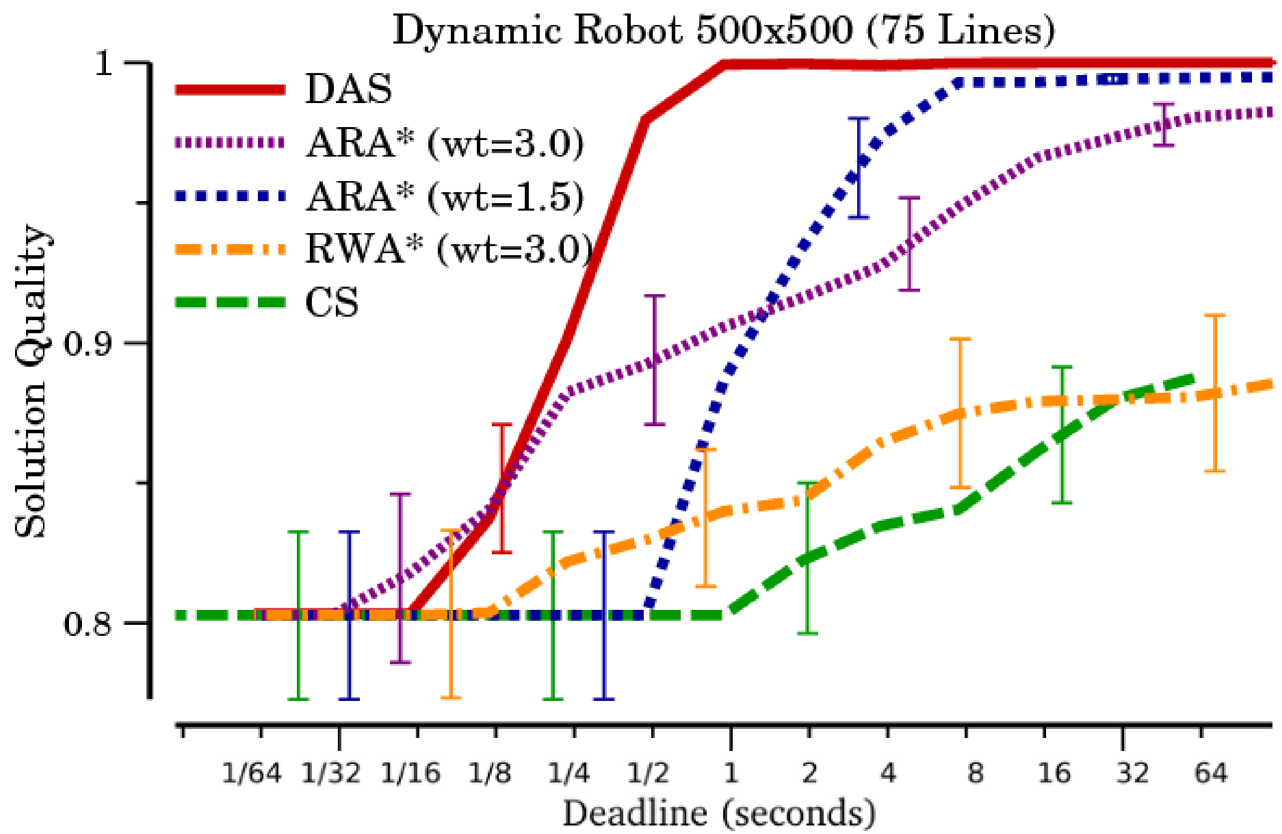
1. while *(time)* $<$ *(deadline)* and *open* is non-empty
2. $\quad d_{max} \leftarrow$ calculate d bound
3. $\quad s \leftarrow$ pop lowest f state from *open*
4. $\quad$ if s is a goal and is better than *incumbent*
5. $\qquad$ *incumbent* $\leftarrow s$
6. $\quad$ else if $\widehat{d}(s) < d_{max}$, expand s
7. $\quad$ else prune s
8. $\quad$ if *open* empties, recover some pruned states
9. return *incumbent*

ripe for improvement!

Results for Deadline-Aware Search



Results for Deadline-Aware Search



Summary So Far

- Optimal search is impractical
- Lots of room for creativity in suboptimal search

Going beyond lower bounds on cost-to-go:

- Inadmissible cost-to-go $\widehat{f}$
- Inadmissible distance-to-go $\widehat{d}$: Speedy
- RR-d uses $f, \widehat{f}$, and $\widehat{d}$
- DAS uses expansion delay

Exploit estimates of uncertainty in beliefs

break?

Suboptimal Search Problem Settings

optimal: minimize solution cost

Suboptimal Search Problem Settings

optimal: minimize solution cost

---

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

tunable: trade cost for time, no guarantees

anytime: incrementally converge to optimal

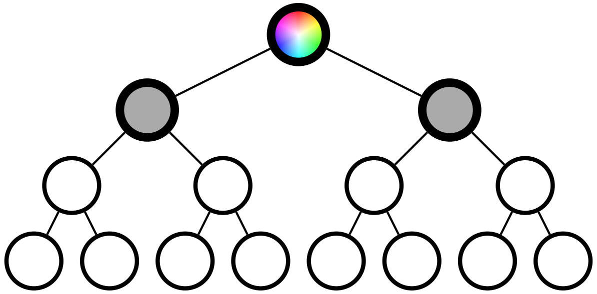
situated: minimize goal achievement time

utility: maximize function of cost and time

real-time: return next action within absolute time bound

Tunable Suboptimal Search

restricted breadth-first search (not best-first!)



Beam search with width 2.

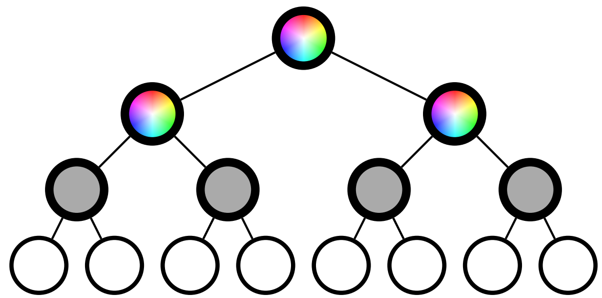
optimal with width $= \infty$, usually faster with smaller k
but no guarantees!
compare comparable nodes, enforce diversity
not forced to fill minima

bead uses d

(see also Steve Wissow's SoCS talk)

Beam Search

restricted breadth-first search (not best-first!)



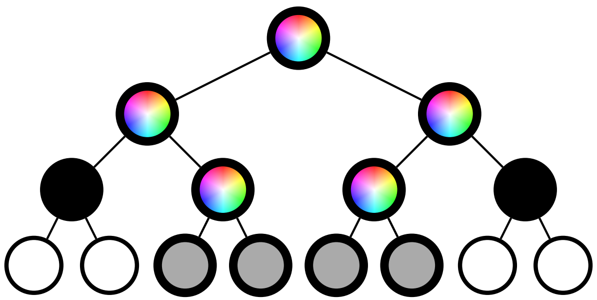
Beam search with width 2.

optimal with width $= \infty$, usually faster with smaller k
but no guarantees!
compare comparable nodes, enforce diversity
not forced to fill minima

bead uses d

(see also Steve Wissow's SoCS talk)

restricted breadth-first search (not best-first!)



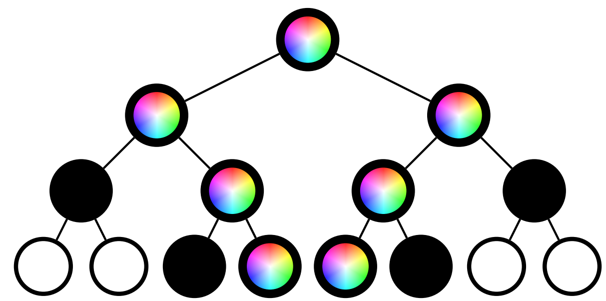
Beam search with width 2.

optimal with width $= \infty$, usually faster with smaller k
but no guarantees!
compare comparable nodes, enforce diversity
not forced to fill minima

bead uses d

(see also Steve Wissow's SoCS talk)

restricted breadth-first search (not best-first!)



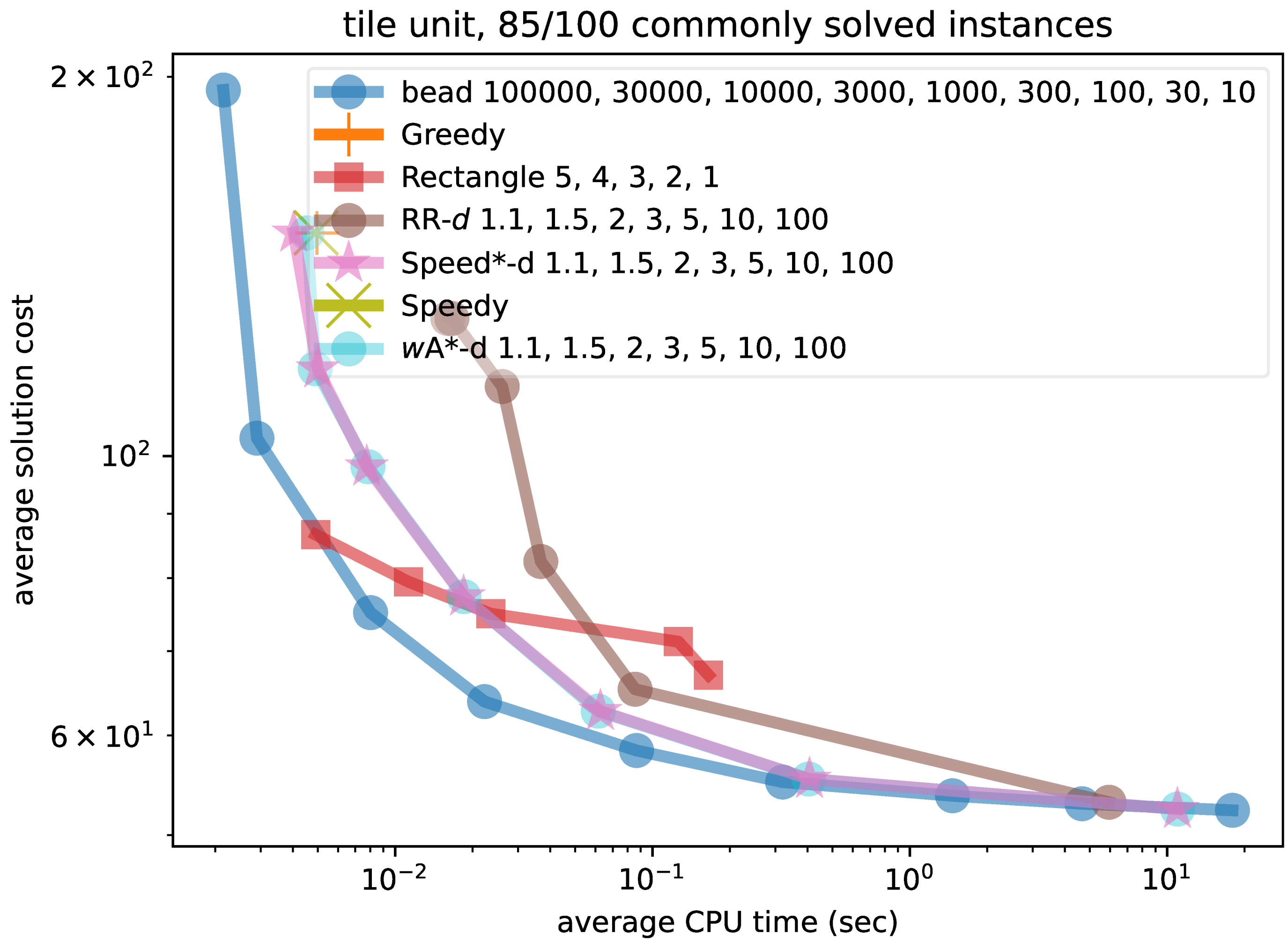
Beam search with width 2.

optimal with width $= \infty$, usually faster with smaller k
but no guarantees!
compare comparable nodes, enforce diversity
not forced to fill minima

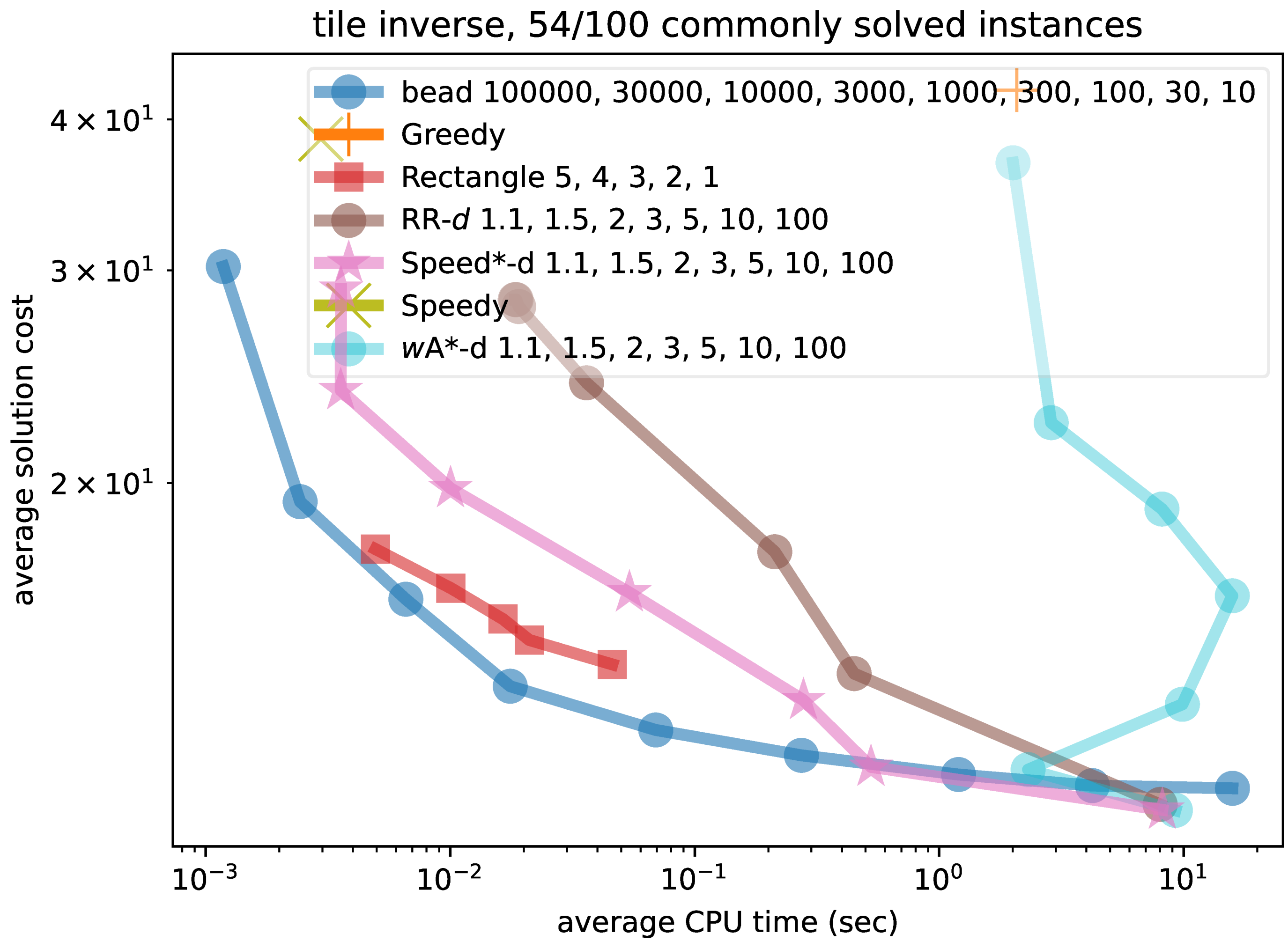
bead uses d

(see also Steve Wissow's SoCS talk)

Bead on Tiles



Bead on Inverse Tiles



Anytime Search

Rectangle Search

iteratively widening and deepening bead

great for large minima!

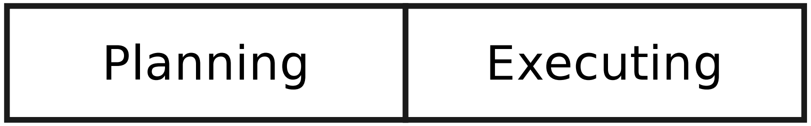
Situated Planning

Achieve goal as quickly as possible!

- Optimal search: slow planning, fast execution

- Greedy search: fast planning, slow execution

- To achieve goal quickly: minimize the sum!



Situated Planning

minimize goal achievement time

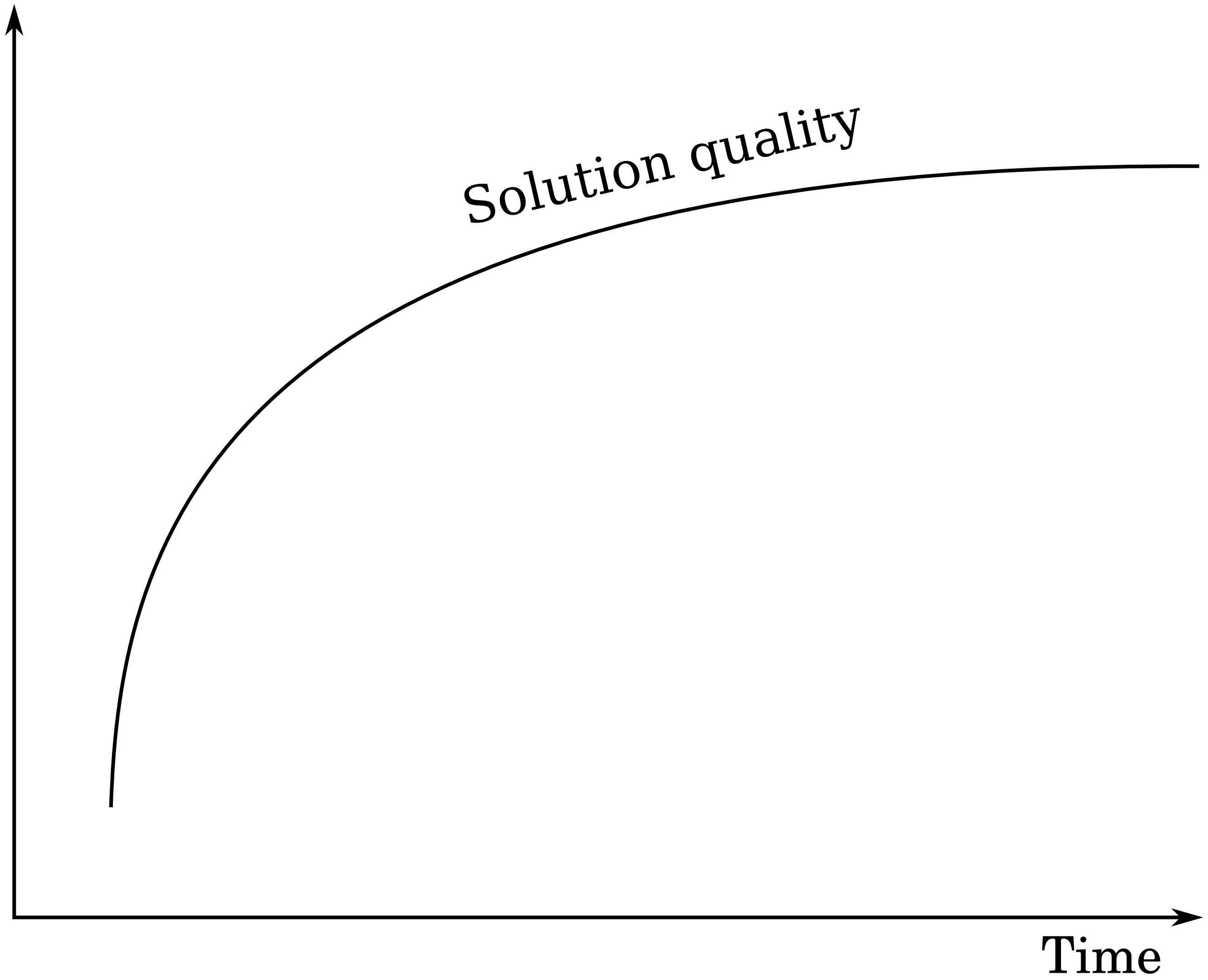
special case of optimizing a utility function of planning time and plan makespan

with concurrent planning and execution: very realistic

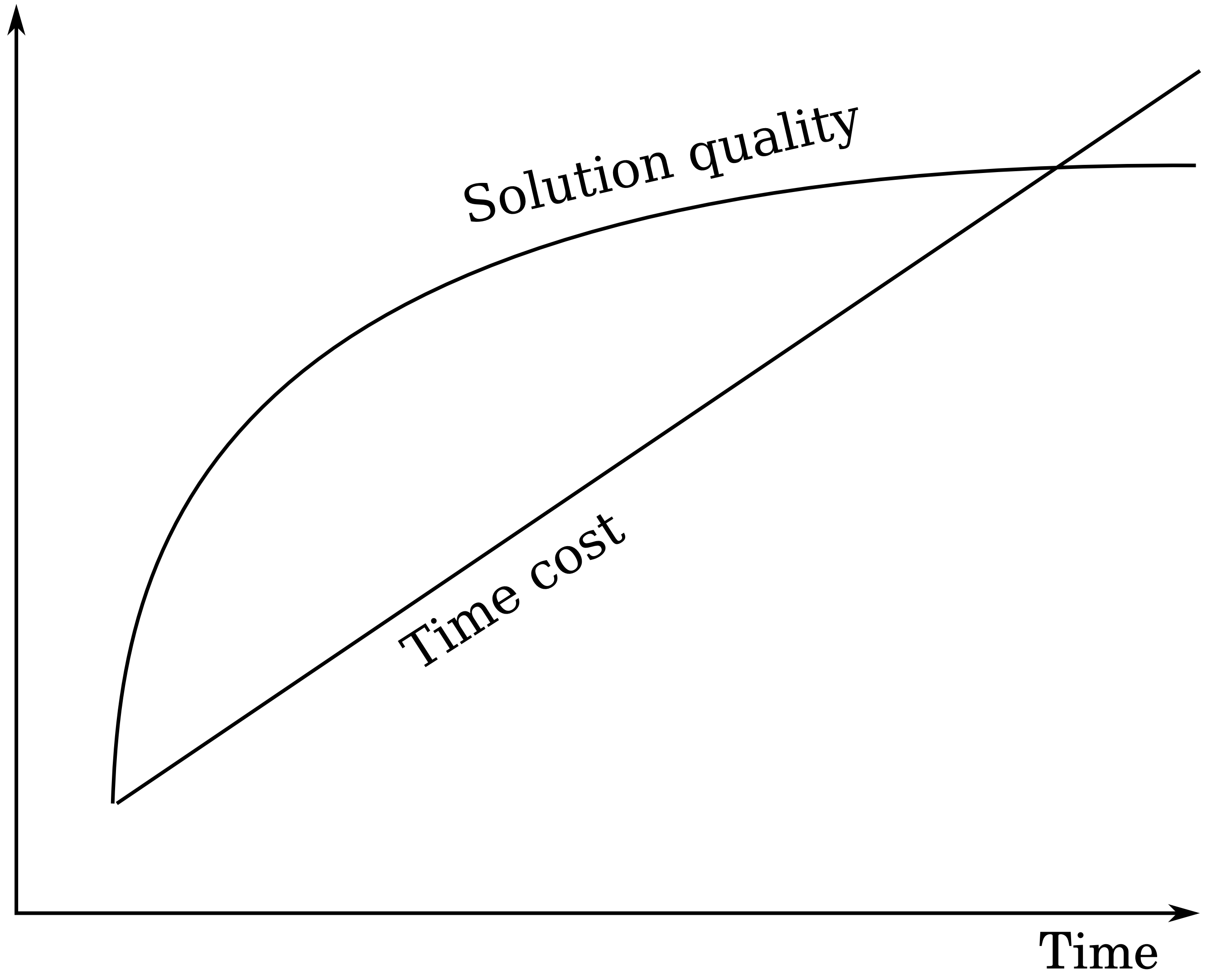
see Coles et al, "Planning and Acting While the Clocks Ticks" at ICAPS!

Utility-based Search

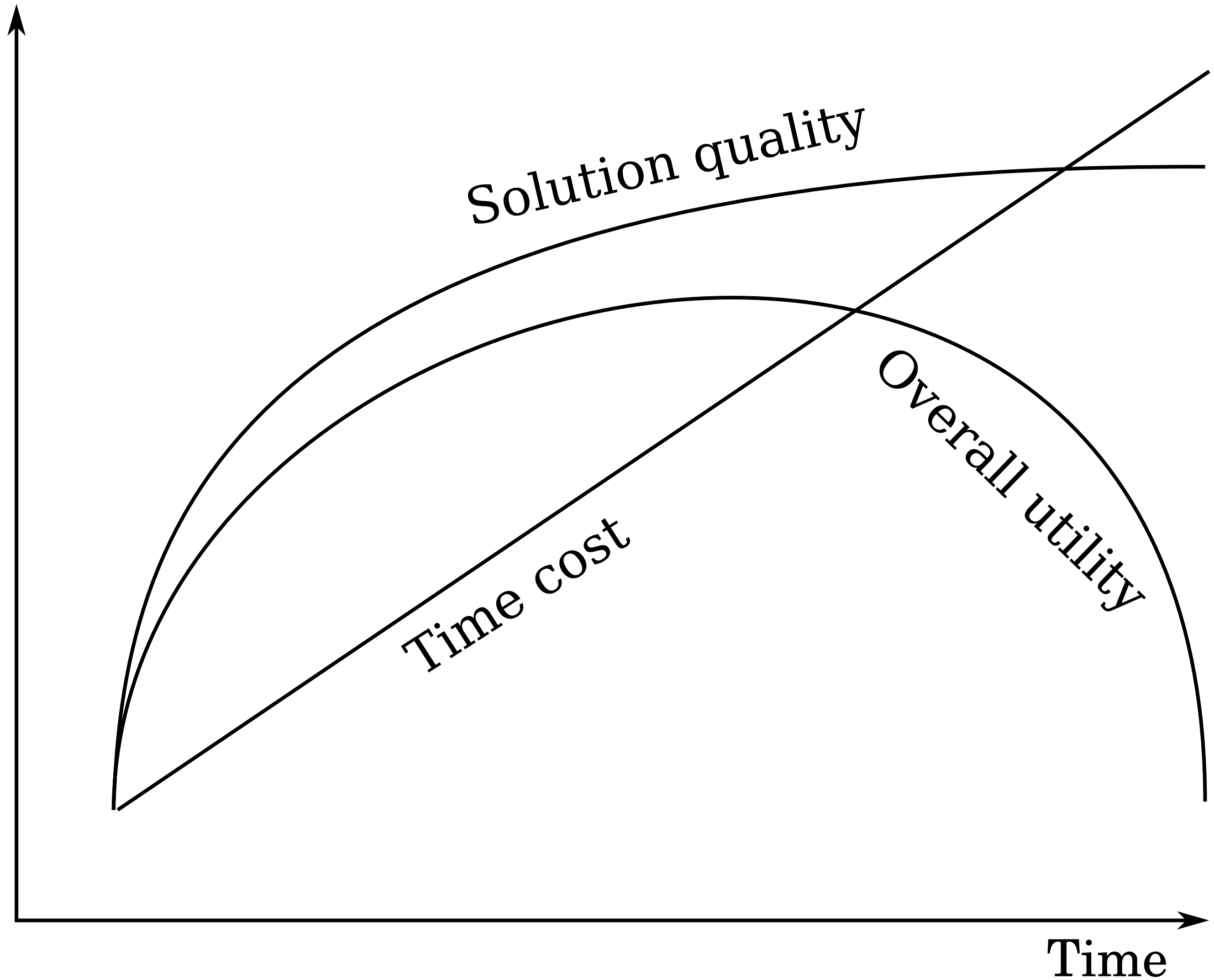
Utility-aware Planning



Utility-aware Planning



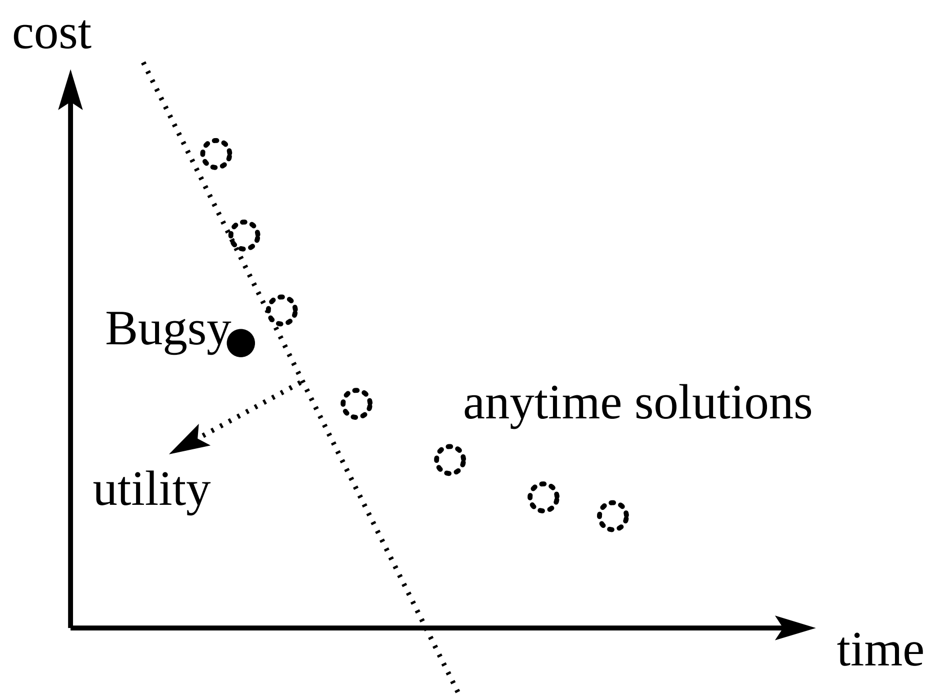
Utility-aware Planning



Utility vs Anytime

Different from anytime algorithms

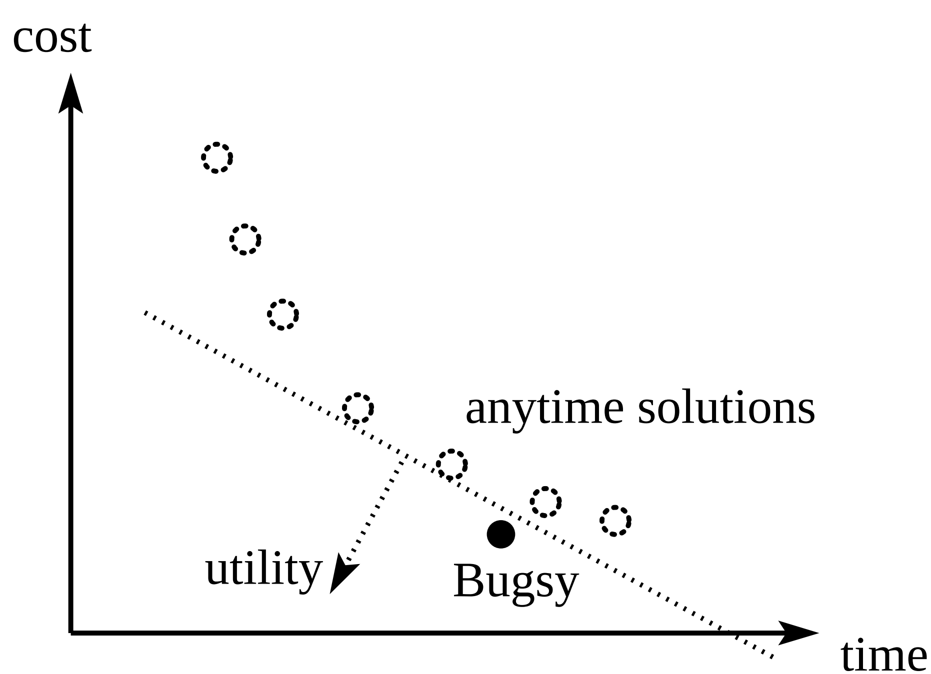
- no need for termination policy, training data
- can spend all effort pursuing one solution
- no fixed trade-off



Utility vs Anytime

Different from anytime algorithms

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



Best-first Utility-guided Search, Yes!

Act to obtain the highest utility outcome.

Possible actions:

terminate: U_{def} = utility of immediately returning null solution

expand(n): estimate utility of outcome enabled by expansion:

$$\hat{u}(n) = -(w_f \cdot \widehat{cost}(n) + w_t \cdot \widehat{time}(n))$$

Expand node with highest $\hat{u}(n)$, unless $< U_{def}$.

Bugsy Details

$$\hat{u}(n) = -(w_f \cdot \widehat{cost}(n) + w_t \cdot \widehat{time}(n))$$

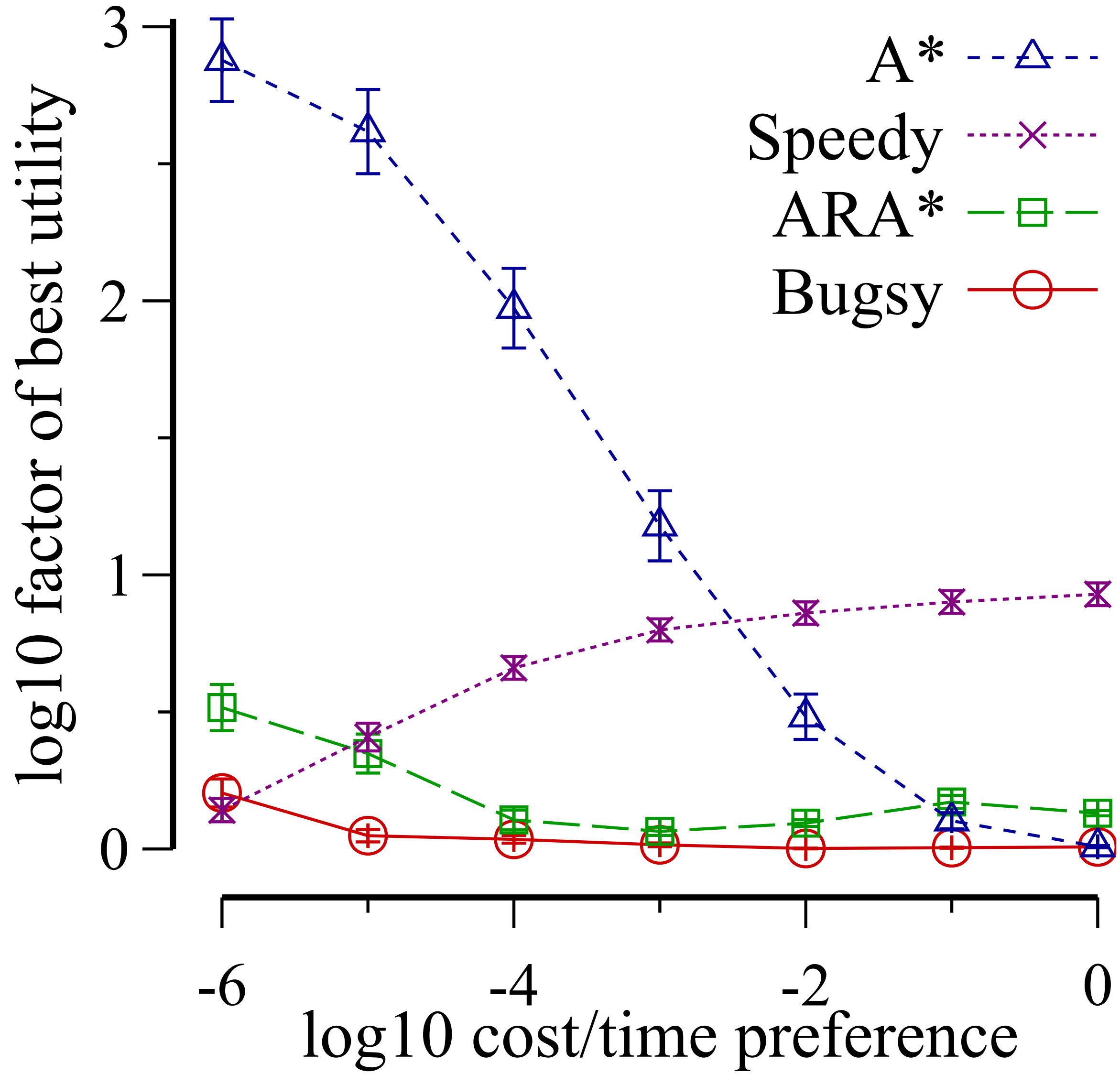
$$\widehat{cost}(n) = f(n)$$

Estimate the amount of vacillation and account for it:

$\hat{d}(n)$ number of search steps
$delay$ number of expansions between steps
t_{expand} time per expansion

$$\widehat{time}(n) = \hat{d}(n) \cdot delay \cdot t_{expand}$$

Bugsy on the 15-puzzle



Multiple Sequence Alignment

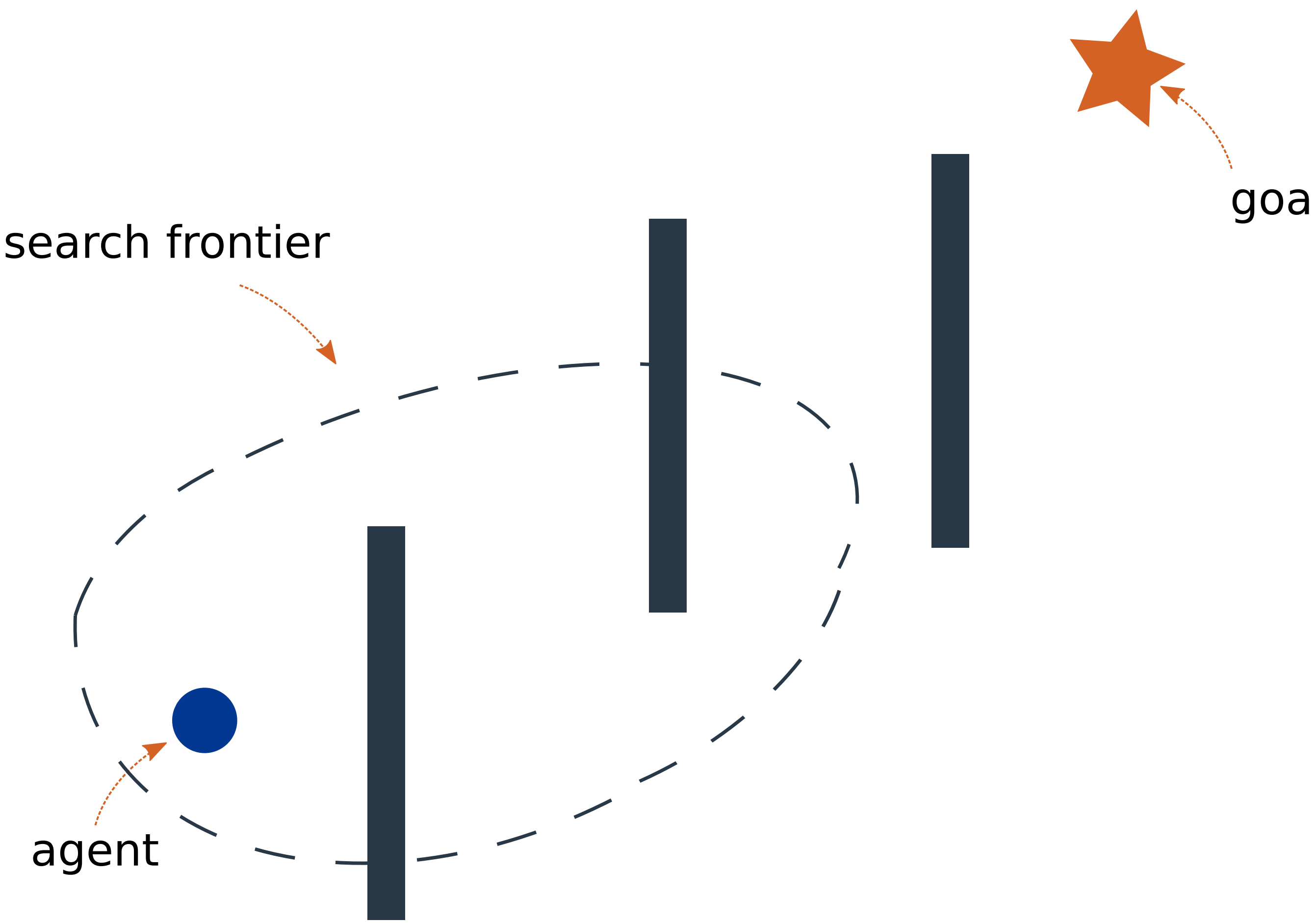
$U()$	Bugsy	ARA*	AA*	Gr	A*
time only	**100**	**100**	**100**	**100**	54
0.1 sec	**99**	97	98	96	54
0.5 sec	**92**	83	88	76	52
1 sec	**80**	68	79	54	51
5 sec	**75**	68	71	25	73
10 secs	**78**	75	74	25	**78**
cost only	**82**	**82**	**82**	24	**82**

Real-time Search

Real-time Heuristic Search

return next action within prespecified time bound

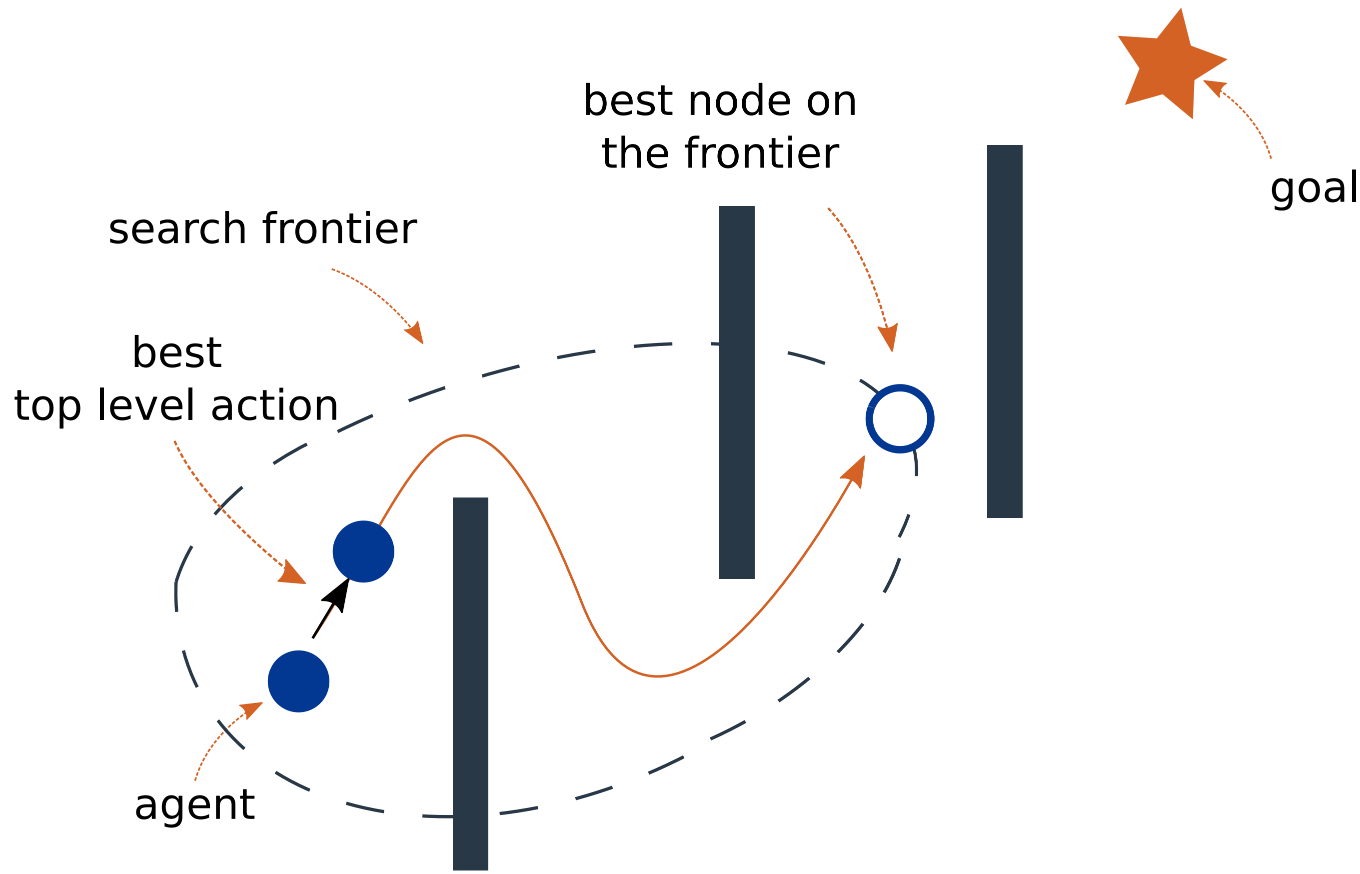
return next action within prespecified time bound



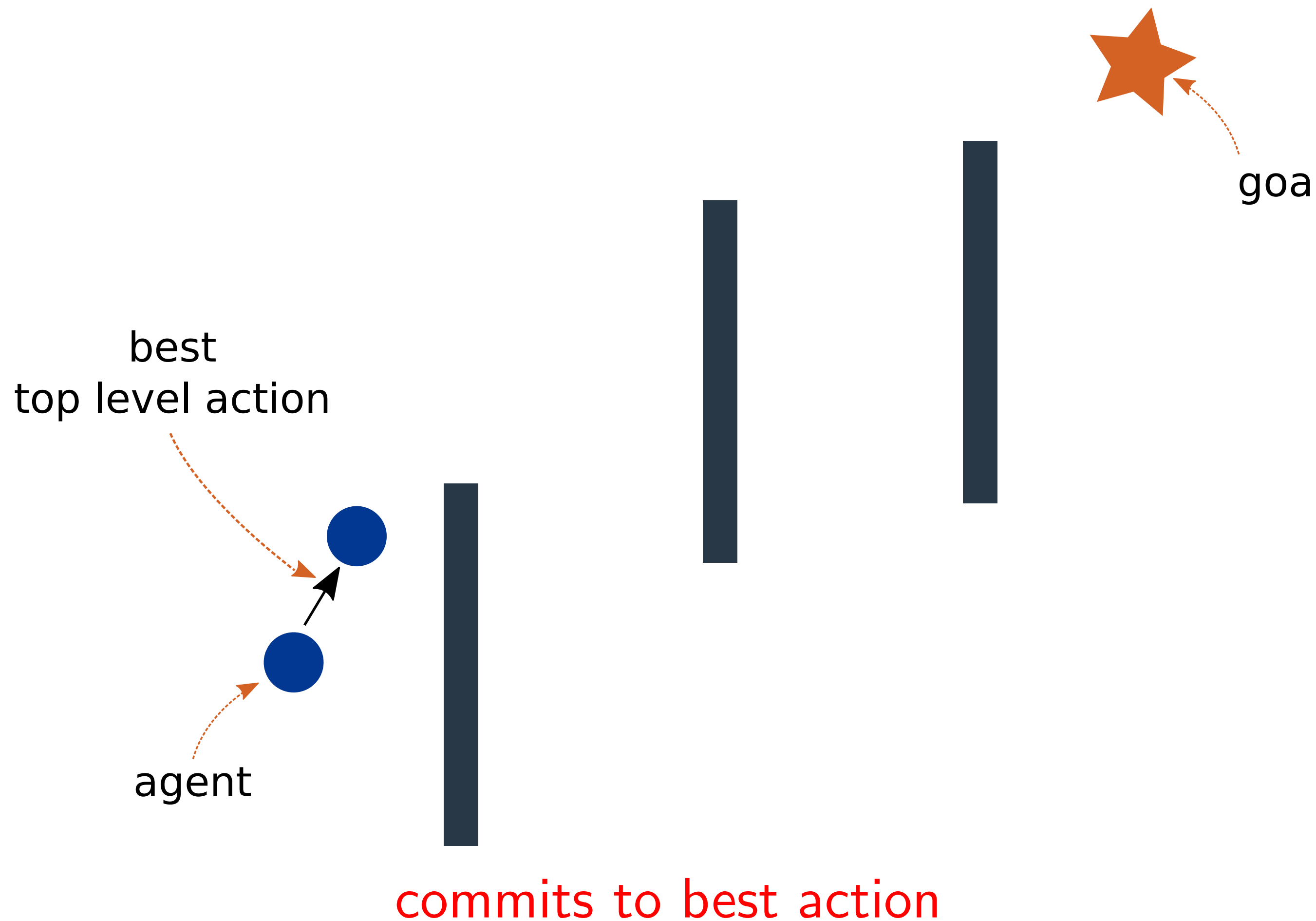
search for bounded time

Real-time Heuristic Search

return next action within prespecified time bound



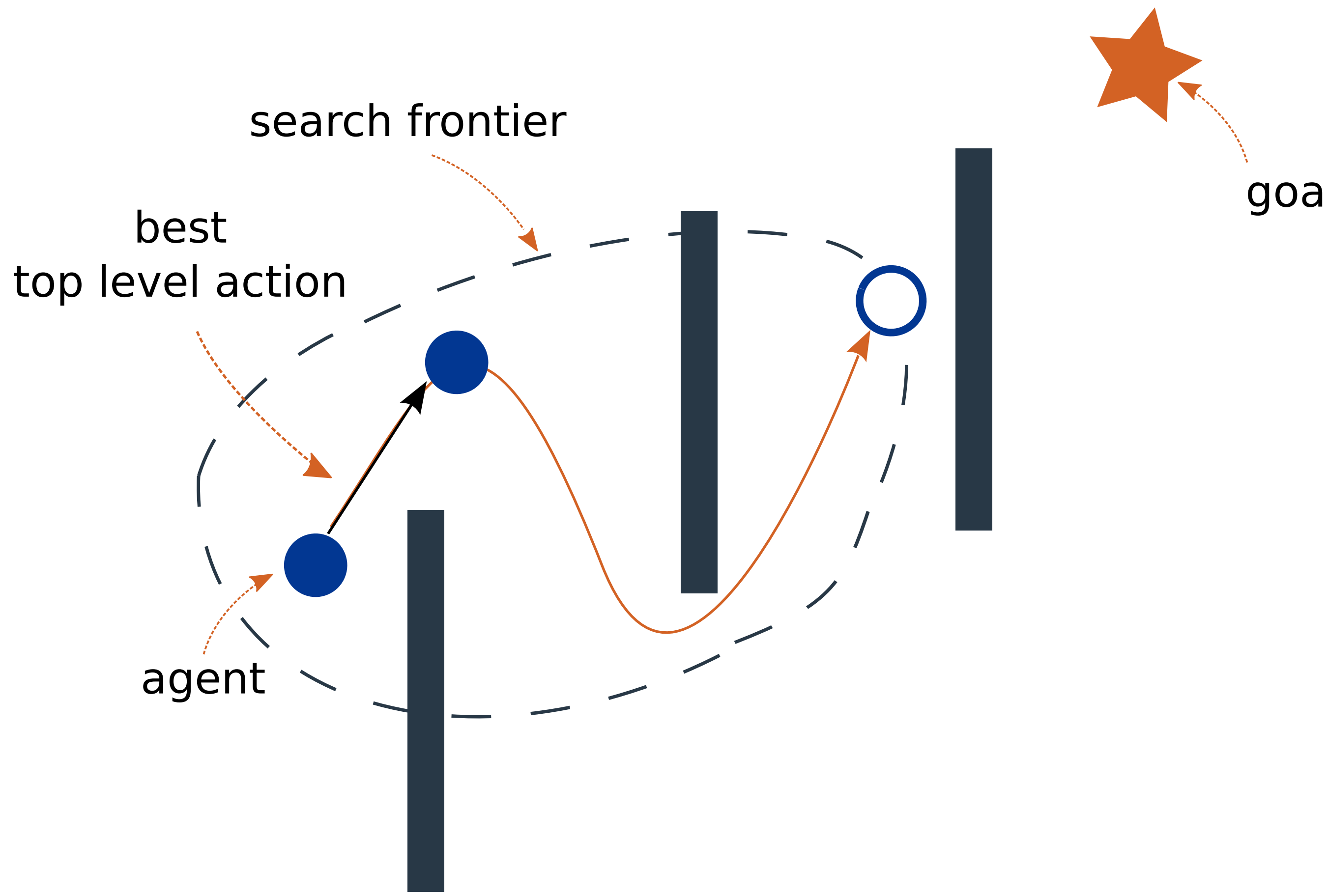
return next action within prespecified time bound



return next action within prespecified time bound

Real-time Heuristic Search

return next action within prespecified time bound



concurrent search and execution, online planning, 'receding horizon control'

Classic Real-time Search: LSS-LRTA* (Koenig&Sun 2008)

three phases:

1. Lookahead:
   expand minimum f node
2. Decision-making:
   backup minimum f from frontier ('minimin')
   select top-level action with minimum f
3. Learning:
   update heuristic values
   (avoid loops, escape local minima, ensure completeness)

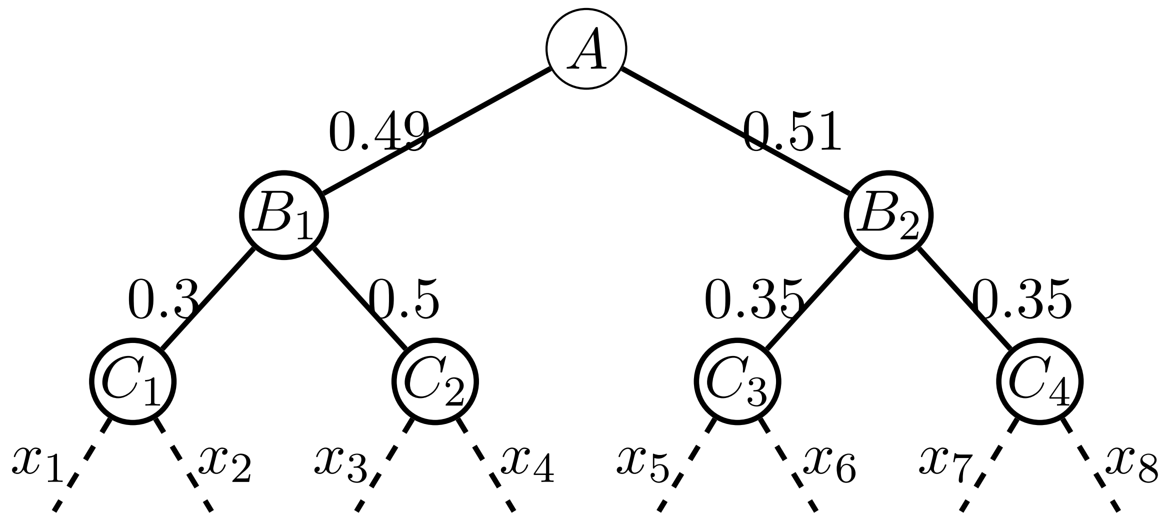
repeat until goal achieved

three phases:

1. Lookahead:
   - Which nodes to expand?
     - minimum f optimal for A* (offline optimal)
     - what about online?
2. Decision-making:
   - Which action to pick?
     - lowest f optimal for A* (offline optimal)
     - what about online?
3. Learning:
   - How to backup from frontier?
     - minimin optimal for A* (offline optimal deterministic)
     - Bellman optimal for VI (offline optimal stochastic)
     - what about online?

Decision-making: An Example (Pemberton & Korf 1995)

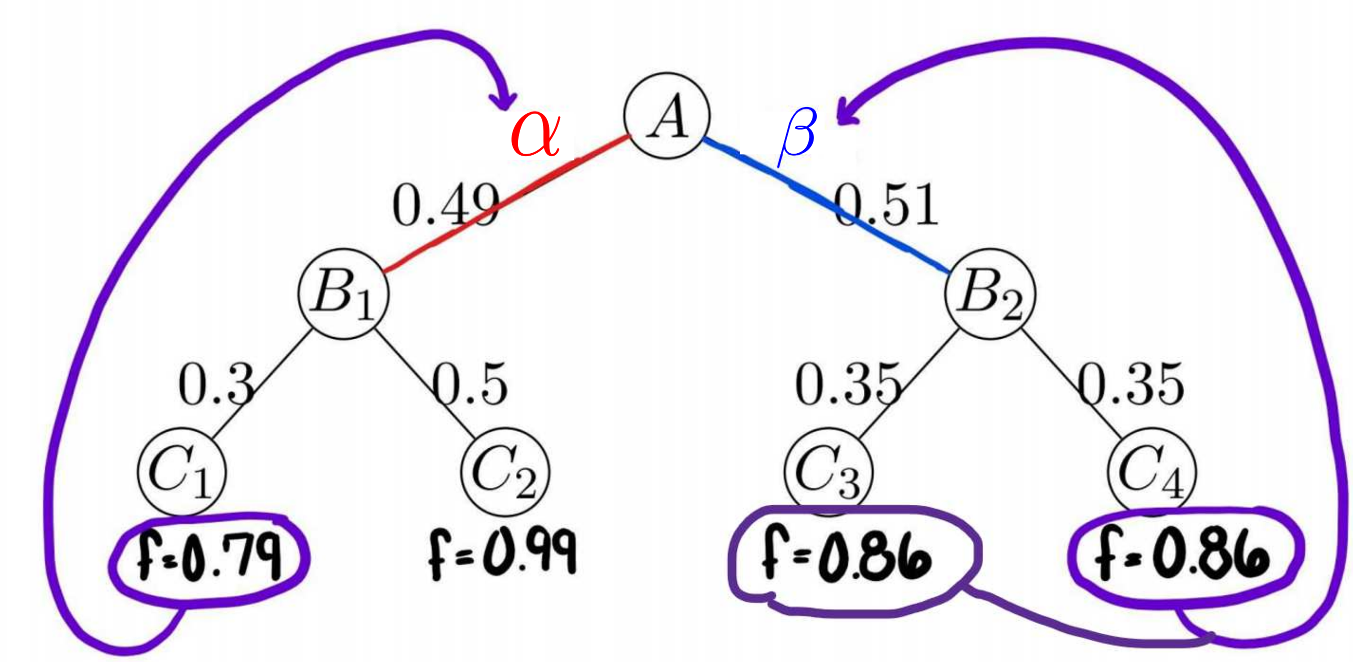
Should an agent at A move to B_1 or B_2?
(x_i are unknown but i.i.d. uniform 0-1)



lower bound on cost-to-go $h = 0$, so $f = g$

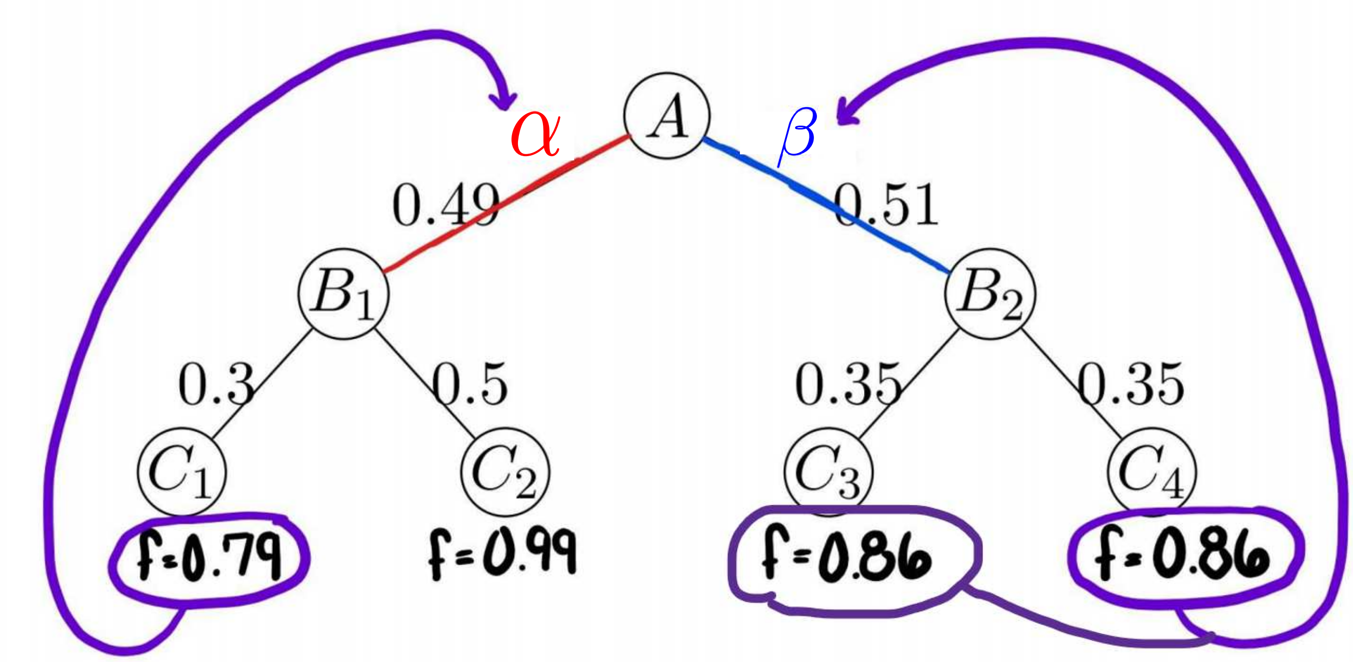
some x_i will be revealed at the next step

Should an agent at A move to B_1 or B_2?
(x_i are unknown but i.i.d. uniform 0-1)



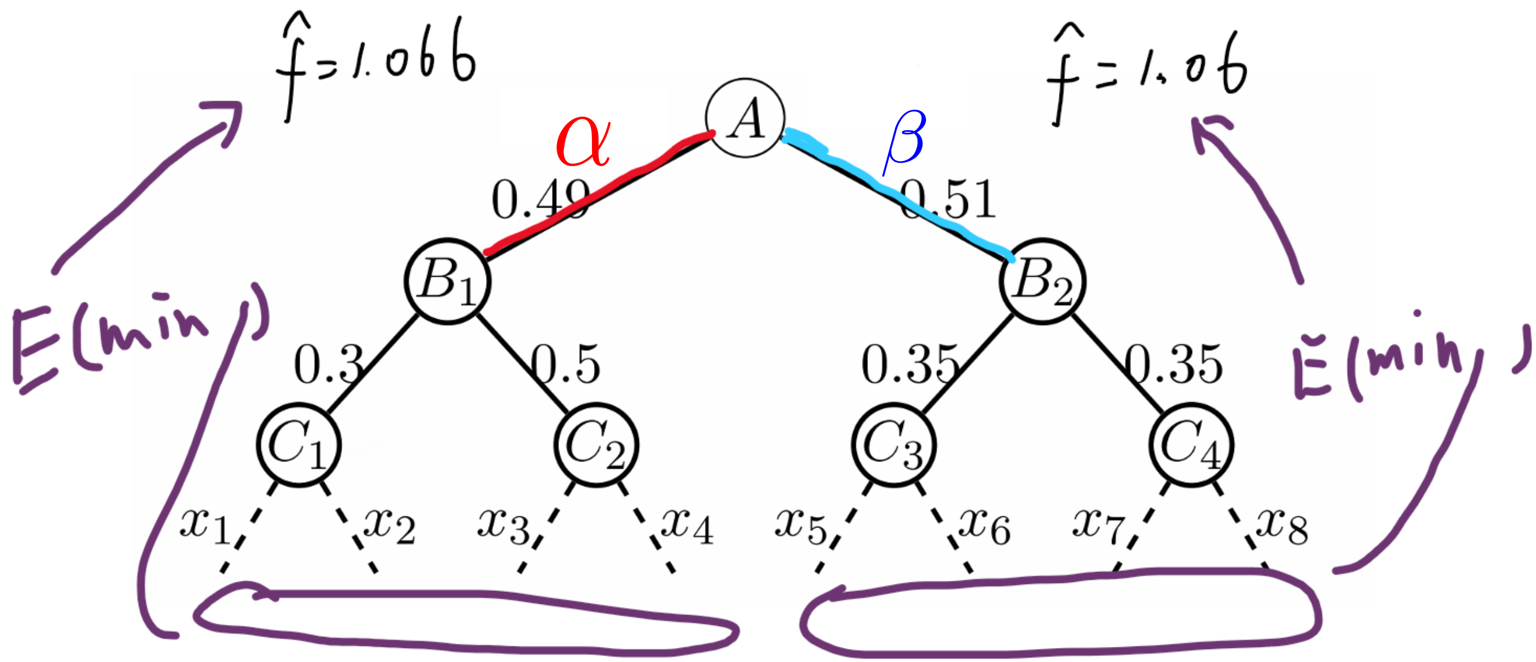
Decision-making: An Example (Pemberton & Korf 1995)

Should an agent at A move to B_1 or B_2?
(x_i are unknown but i.i.d. uniform 0-1)



but decision theory says minimize expected value

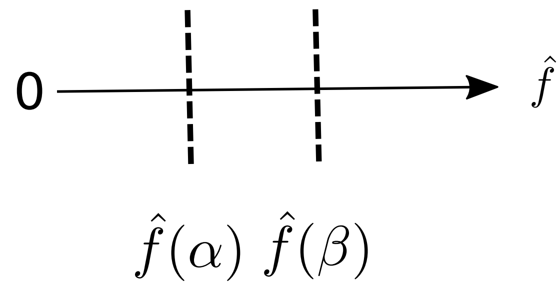
Should an agent at A move to B_1 or B_2?
(x_i are unknown but i.i.d. uniform 0-1)



$\hat{f}$ is expected total plan cost

four x_i will be revealed at the next step

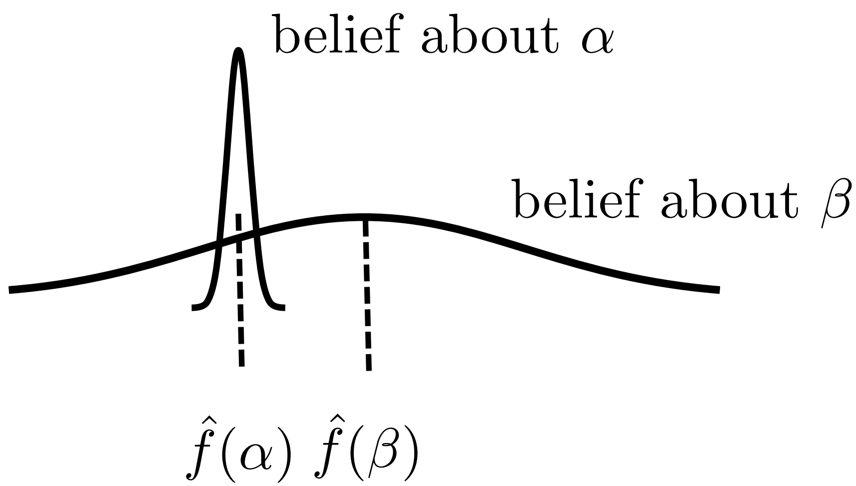
Lookahead: An Example



$\widehat{f}$ is expected value

Should an agent expand nodes under α or β?

Lookahead: An Example



$\widehat{f}$ is expected value

Should an agent expand nodes under α or β?

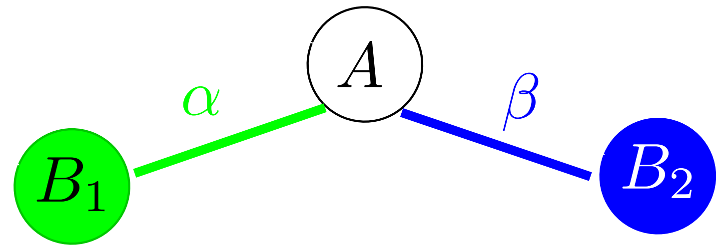
$\widehat{f}$ is not the answer: what to do?
want to maximize value of information
need to consider uncertainty of estimates

Nancy (Mitchell et al, AAAI-19; Fickert et al, AAAI-20)

- want to maximize value of information
- expand nodes which minimize expected regret
- relies on belief over values
- choose expansions that decrease uncertainty about best

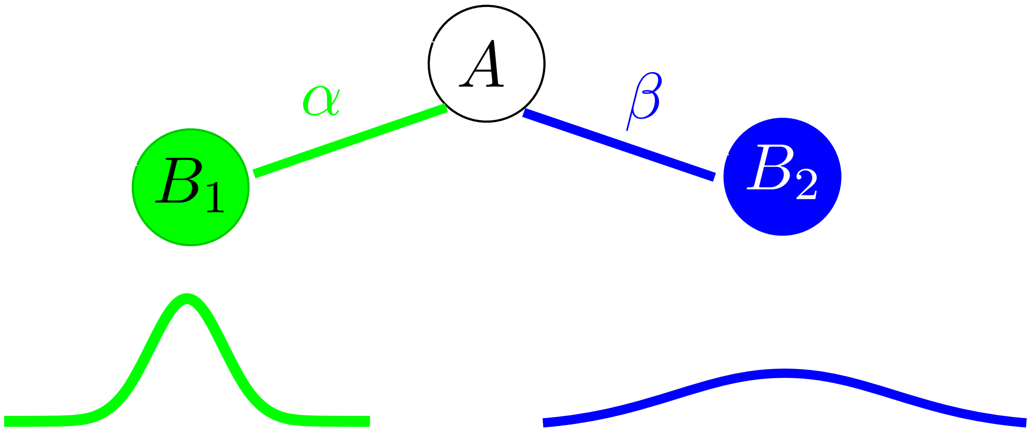
Risk-based Lookahead Example

expand under α or β?



Risk-based Lookahead Example

expand under α or β?

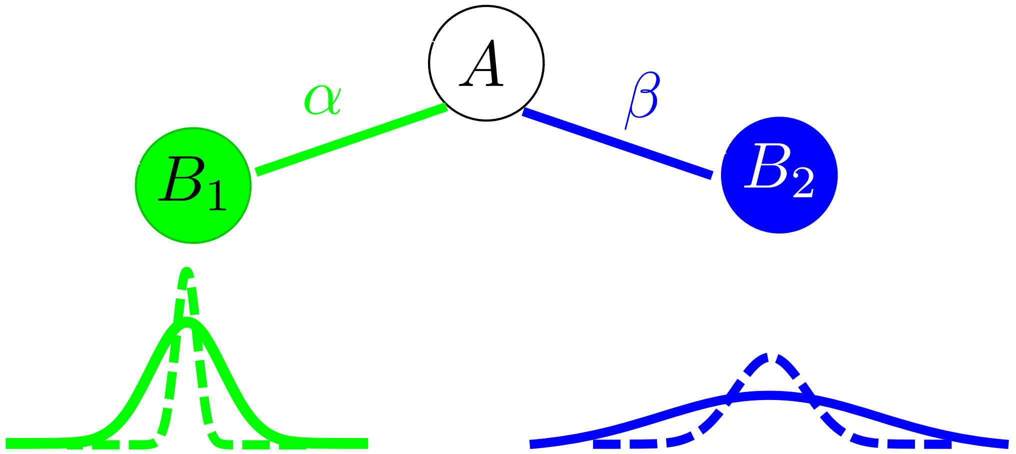


need 2 things:

1) current beliefs
2) estimate of how beliefs might change with search

Risk-based Lookahead Example

expand under α or β?

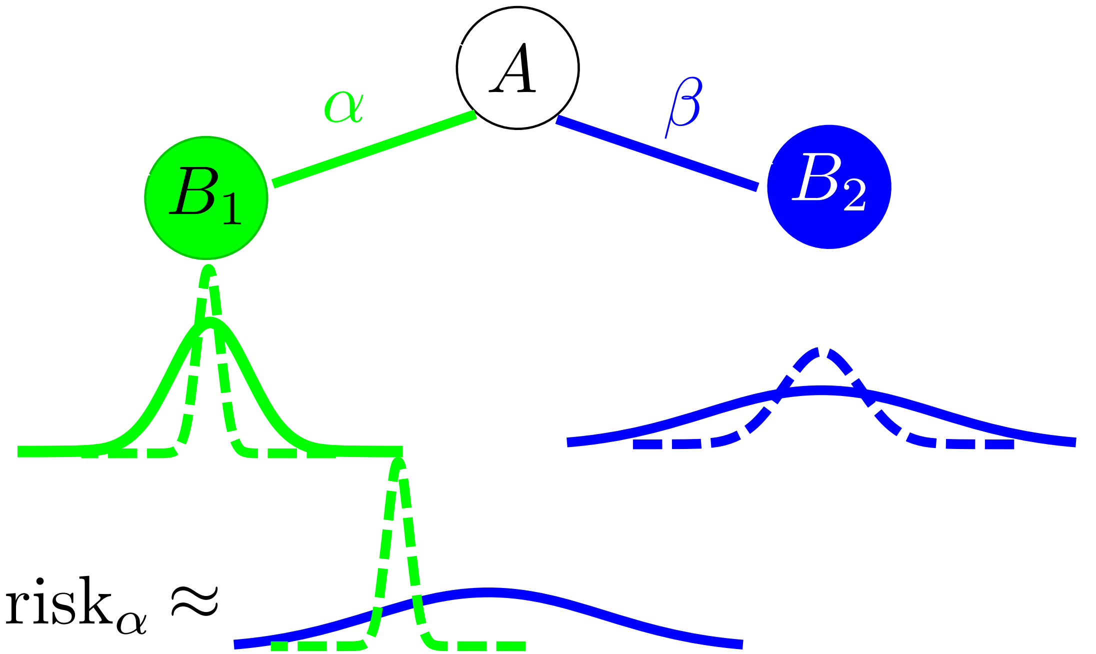


need 2 things:

1) current beliefs
2) estimate of how beliefs might change with search

Risk-based Lookahead Example

expand under α or β?

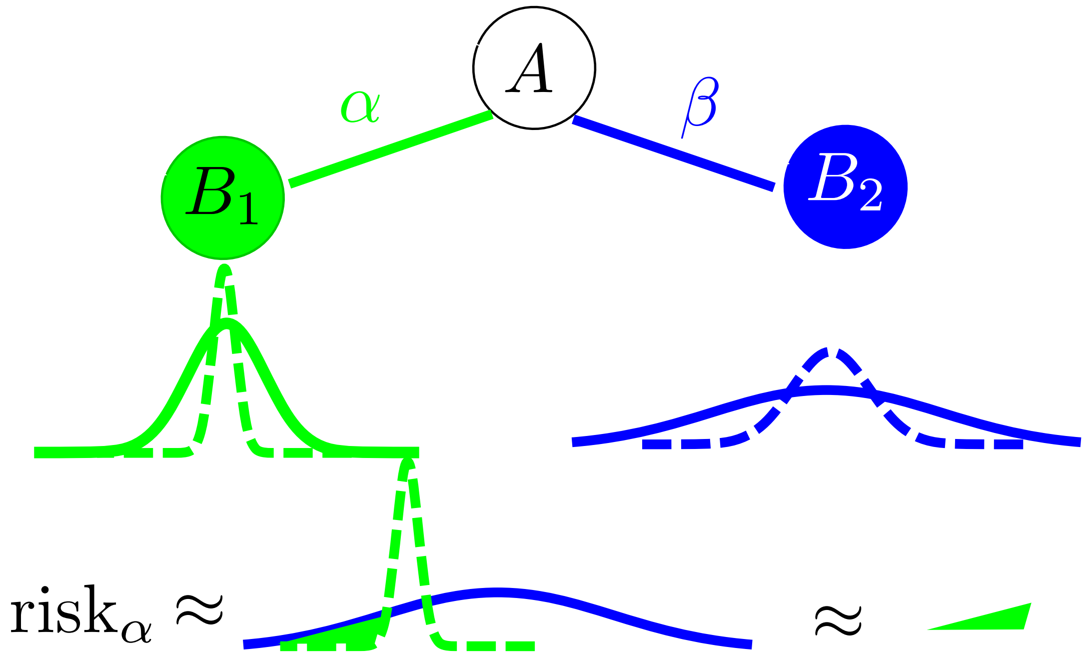


Risk: expected regret if a suboptimal action is selected
α is TLA with lowest expected value, other is β

$$\mathbb{E}\left[\underbrace{f^*(\alpha) - f^*(\beta)}_{\text{our regret}} \;\middle|\; \underbrace{f^*(\beta) < f^*(\alpha)}_{\text{when } \alpha \text{ not best}}\right]$$

Risk-based Lookahead Example

expand under α or β?



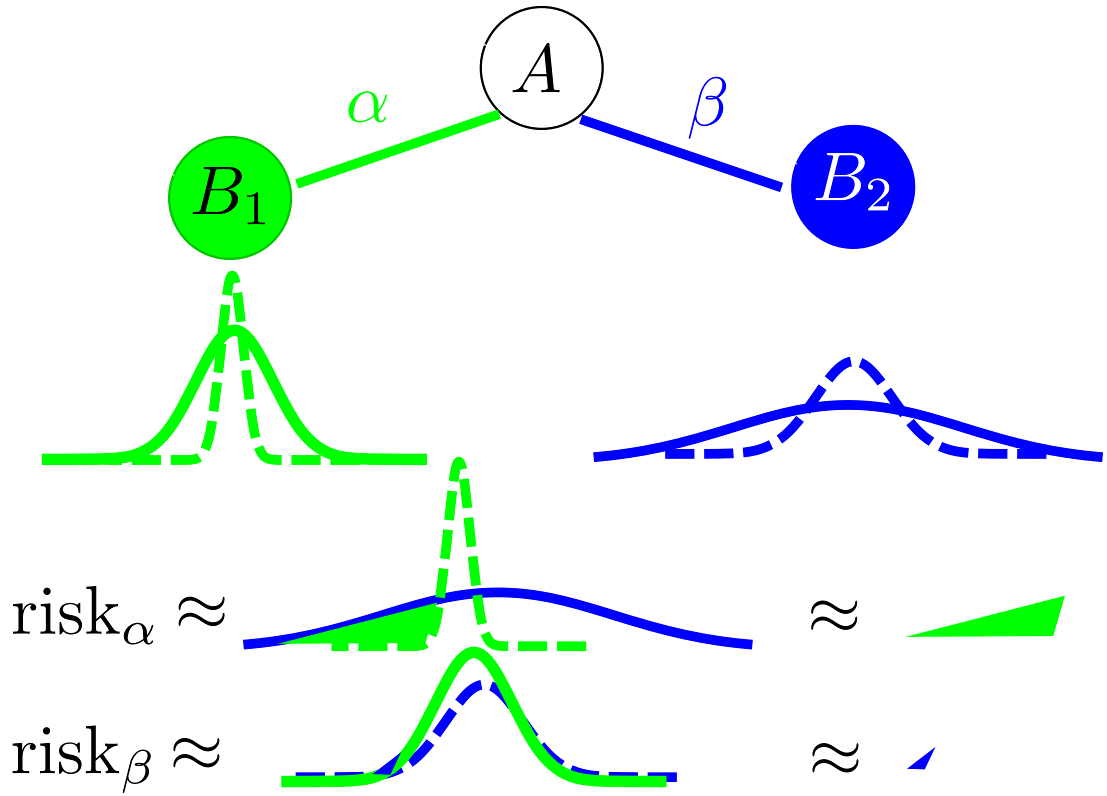
Risk: expected regret if a suboptimal action is selected

α is TLA with lowest expected value, other is β

$$\mathbb{E}\left[\underbrace{f^*(\alpha) - f^*(\beta)}_{\text{our regret}} \;\middle|\; \underbrace{f^*(\beta) < f^*(\alpha)}_{\text{when } \alpha \text{ not best}}\right]$$

Risk-based Lookahead Example

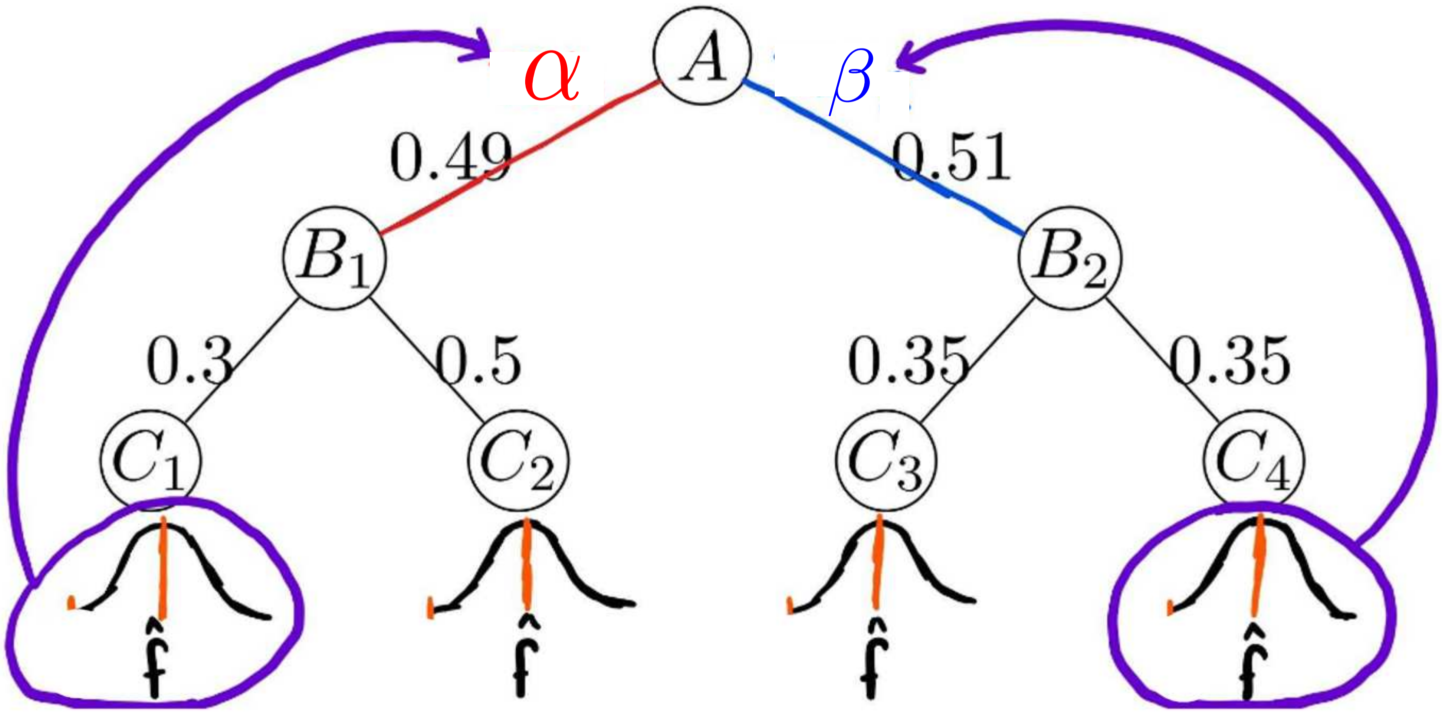
expand under α or β?



expand under the TLA that minimizes risk!

expand under β!

Nancy Backups



Nancy:
parent $\leftarrow$ belief with minimum $\widehat{f}$ among successors
conveys an entire belief distribution

Open Problem: How to Back-up Frontier Values

minimin: parent gets best child's f

assumes no more information will become available

Bellman: parent gets expected-best child's $\widehat{f}$

assumes no more information will become available

Nancy: parent gets expected-best child's belief

assumes no more information will become available

Cserna: parent gets expected min over all children's beliefs

assumes we will know optimal choices

something intermediate would seem appropriate!

Nancy's Response to the Central Issues

1. Lookahead:
   Which nodes to expand?
   those that minimize risk
2. Decision-making:
   Which action to pick?
   minimum $\widehat{f}$ (rationality)
3. Learning:
   How to backup from frontier?
   backup beliefs ('Nancy backups')

minimizing uncertainty drives the search

see also XES (bounded-cost search, IJCAI-21)

How to Form Beliefs?

Nancy: Heuristic values: scalar $\rightarrow$ probability distribution (belief)

How to form beliefs?

How to Form Beliefs?

Nancy: Heuristic values: scalar $\rightarrow$ probability distribution (belief)

How to form beliefs?

assumptions:

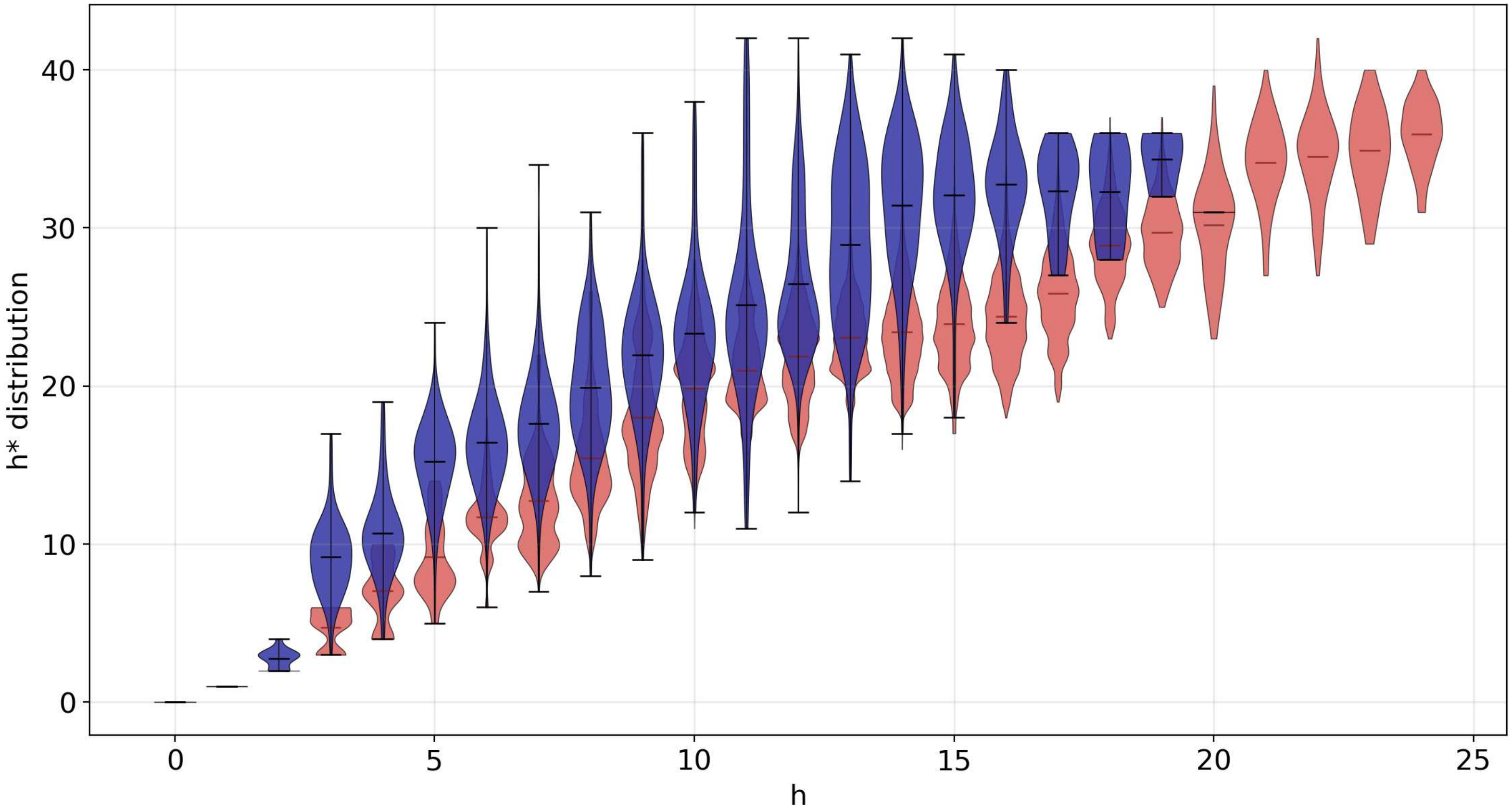
Gaussian at $\widehat{f}$ with width $\propto \widehat{d}$, truncated at f
online learning with few parameters

training data:

histogram of previous h^* given h
offline learning with many parameters

Example h^* Distribution: Transport vs Blocks World

What do the distributions look like?



Beliefs differ by domain. Often not Gaussian!

A New General Completeness Proof

conditions on problem:

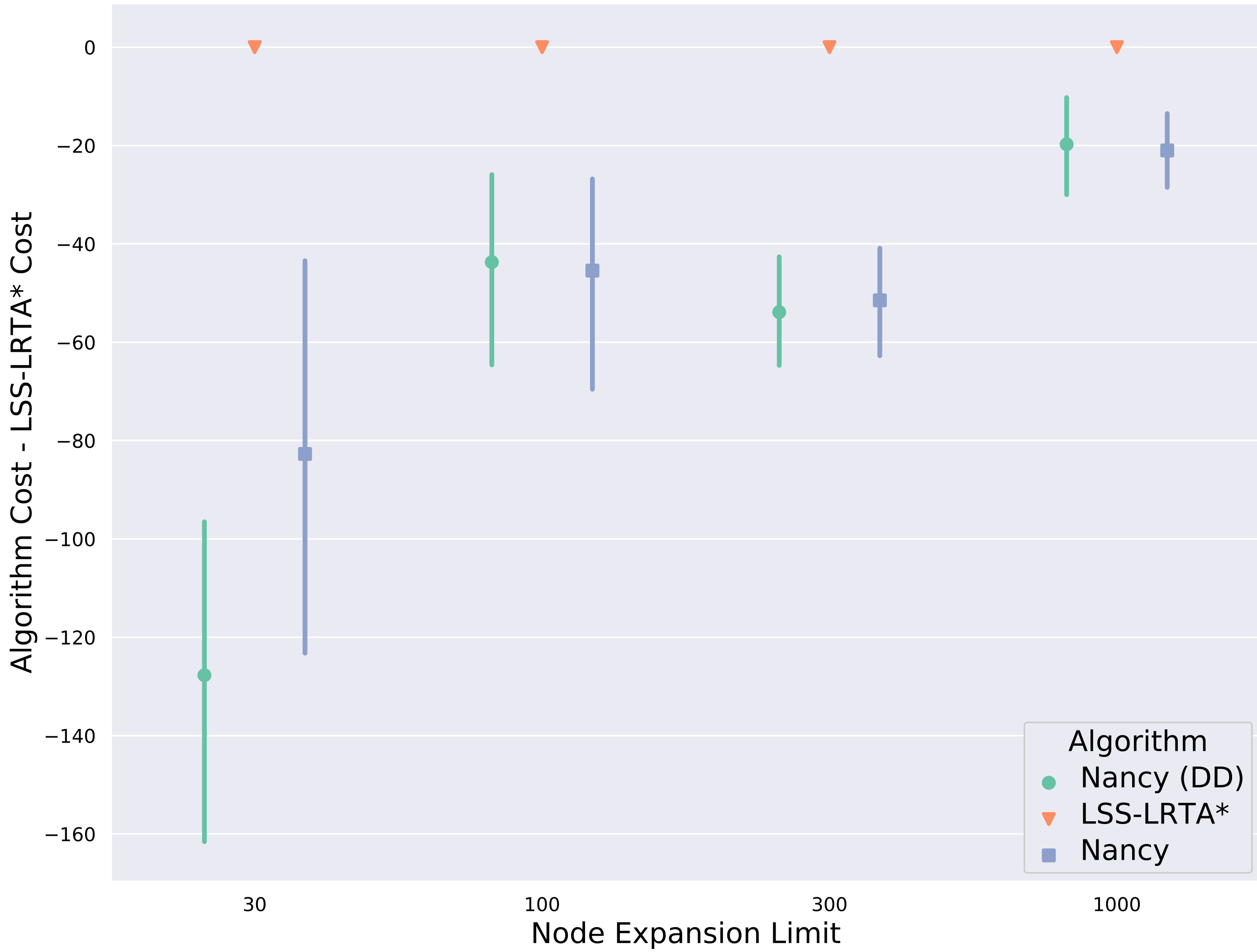
1. initial beliefs have finite expected value
2. positive action costs
3. finite state space
4. no dead-ends

conditions on algorithm:

1. goal-aware
2. learning creates local consistency (eg, DP)
3. selects actions via $\widehat{f}$

This proof applies to any LSS-LRTA*-style algorithm

Example Results: Racetrack



Even assumptions work well!

IPC Planning: Mean Solution Cost

	LSS-LRTA*	Nancy (Gauss)	Nancy (data)
Barman	559	702	**415**
Blocksworld	35	39	**34**
Elevators-unit	34	27	**26**
Parking	62	**27**	31
Rovers	31	**29**	33
Satellite	**15**	17	16
Termes	662	**129**	238
Tidybot	30	30	**29**
Transport	499	567	**422**
Transport-unit	35	29	**27**
VisitAll	52	**50**	52

Data works when assumptions don't!

Conclusion

Recap: Suboptimal Heuristic Search

Suboptimal search is the most important kind!

- practical instead of provably intractable
- distinct settings: bounded-suboptimal, contract, utility...

many sources of information beyond h!

- unbiased estimates (can be learned online)
- distance-to-go, not just cost
- beliefs can model uncertainty

search algorithm as agents

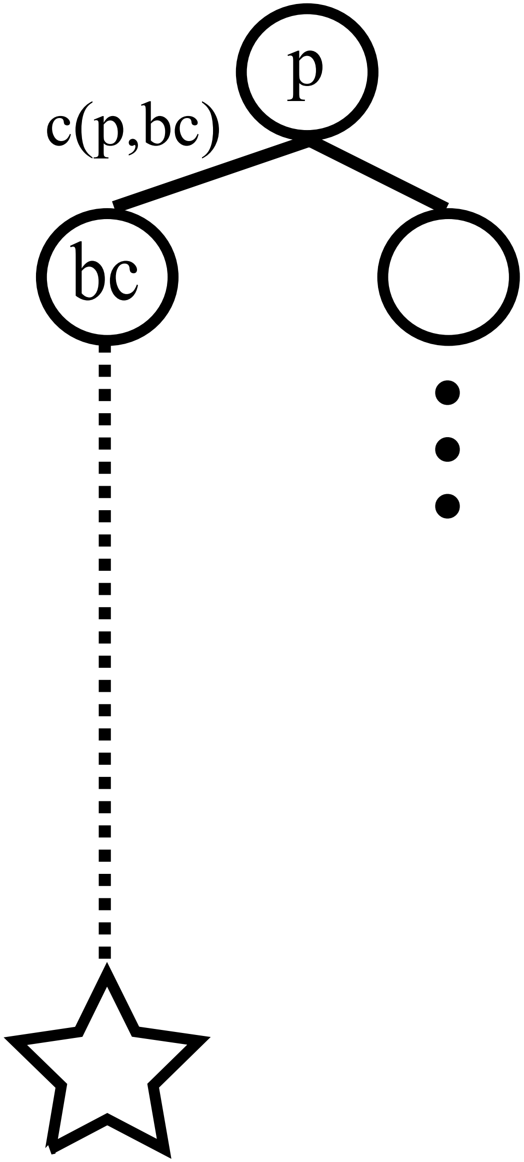
- entire AI agent toolbox applies
  - what to represent, how to estimate
  - how to exploit experience
- search highlights issues more clearly than RL

Suboptimal heuristic search needs YOU!

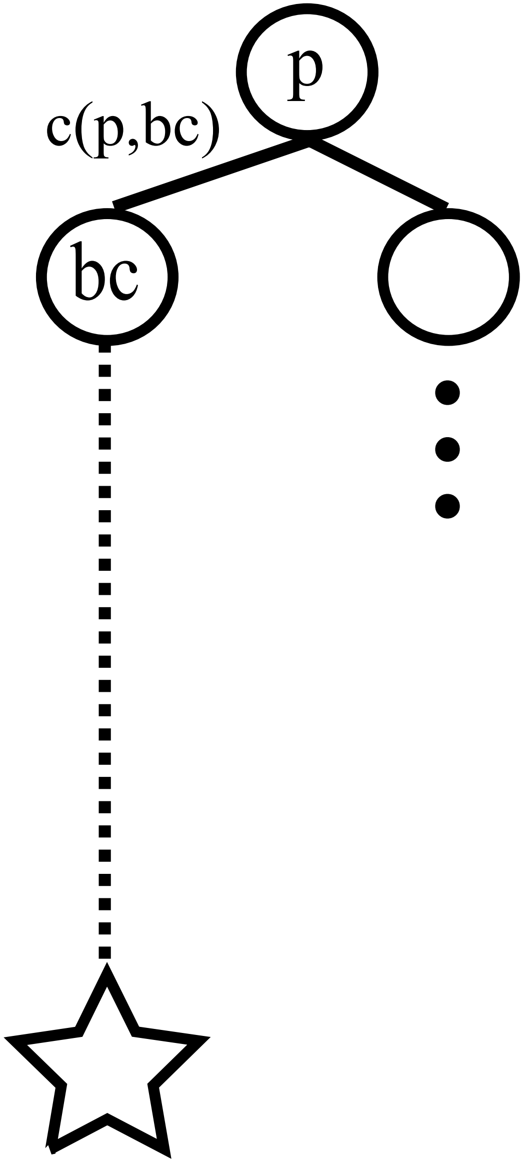
Questions?

Back-Up Slides

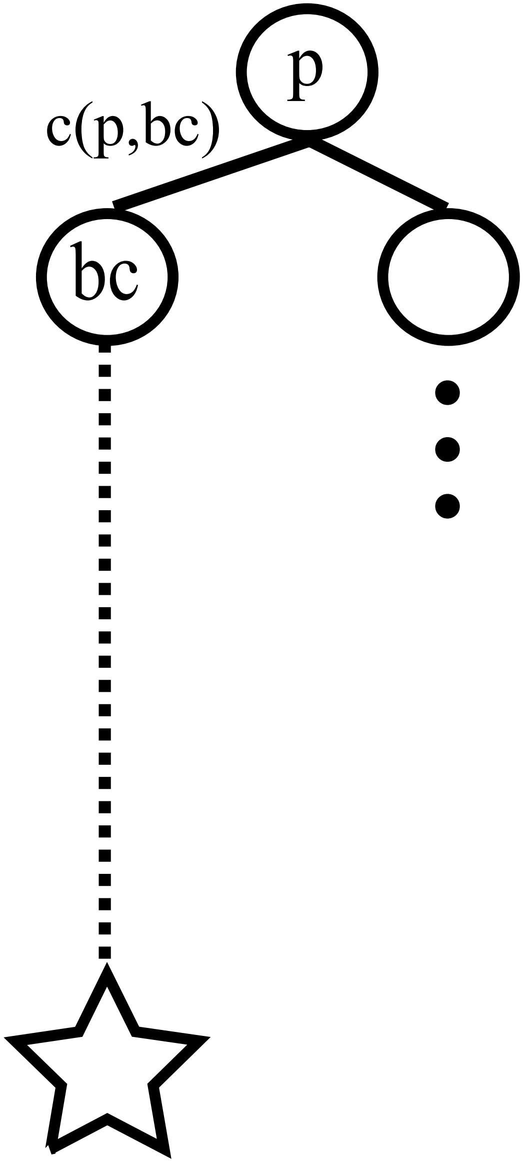
Debiasing h Via Temporal Difference Learning



Debiasing h Via Temporal Difference Learning



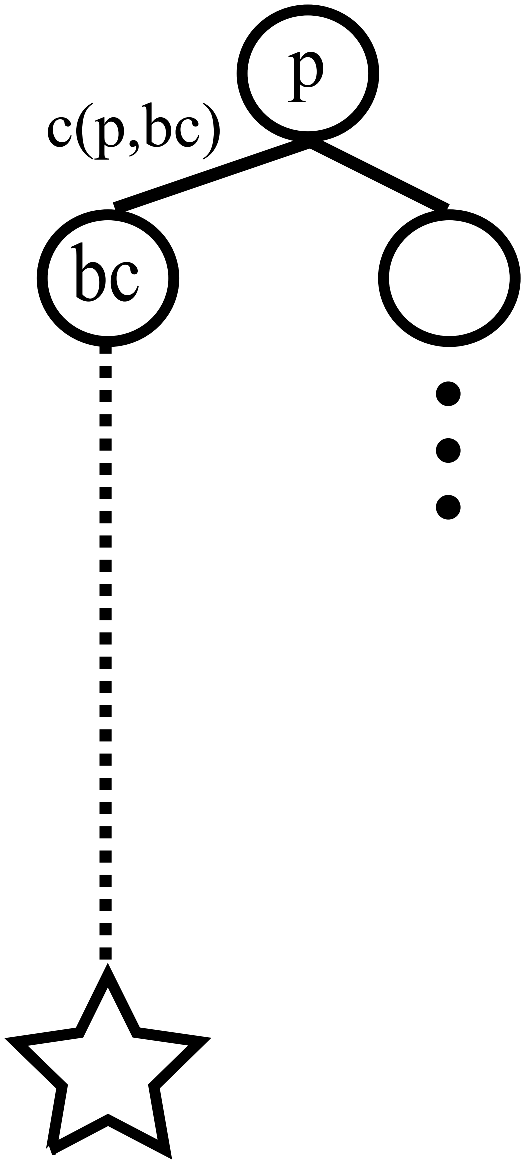
$f(p)$ should equal $f(bc)$



$f(p)$ should equal $f(bc)$

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

Debiasing h Via Temporal Difference Learning

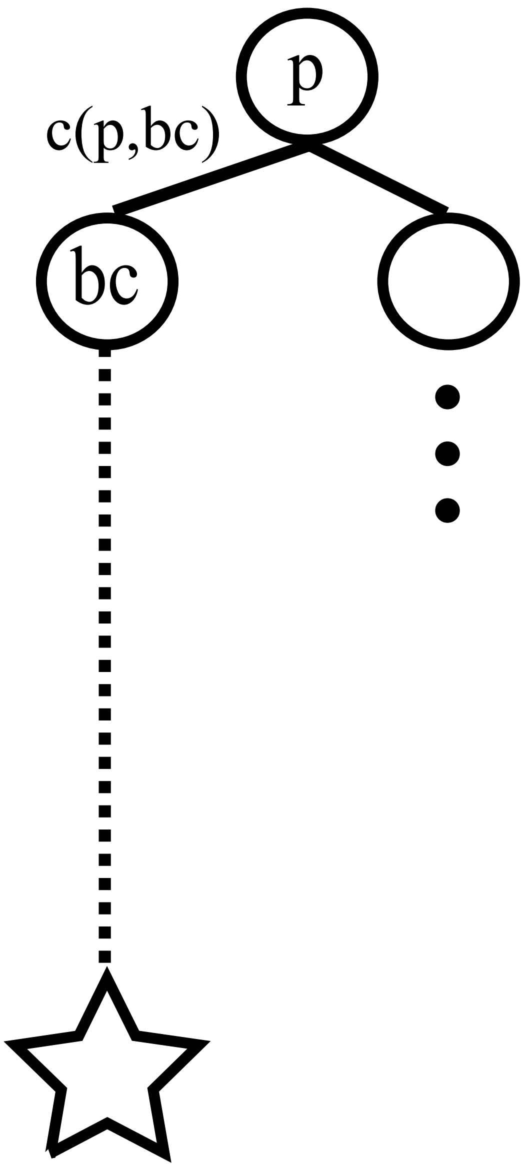


$f(p)$ should equal $f(bc)$

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

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

Debiasing h Via Temporal Difference Learning



$f(p)$ should equal $f(bc)$

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

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

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

EES' Bounded Suboptimality

$$g(n) = \begin{array}{c} f(n) \le w \cdot f_{min} \\ \end{array} \le w \cdot f^*$$

A Science of Suboptimal Search

it's time to take suboptimality seriously!

- estimates, not lower bounds
- belief distributions to quantify uncertainty

acting under uncertainty to maximize utility it's time to take suboptimality seriously!

- estimates, not lower bounds
- belief distributions to quantify uncertainty

acting under uncertainty to maximize utility
= all of AI