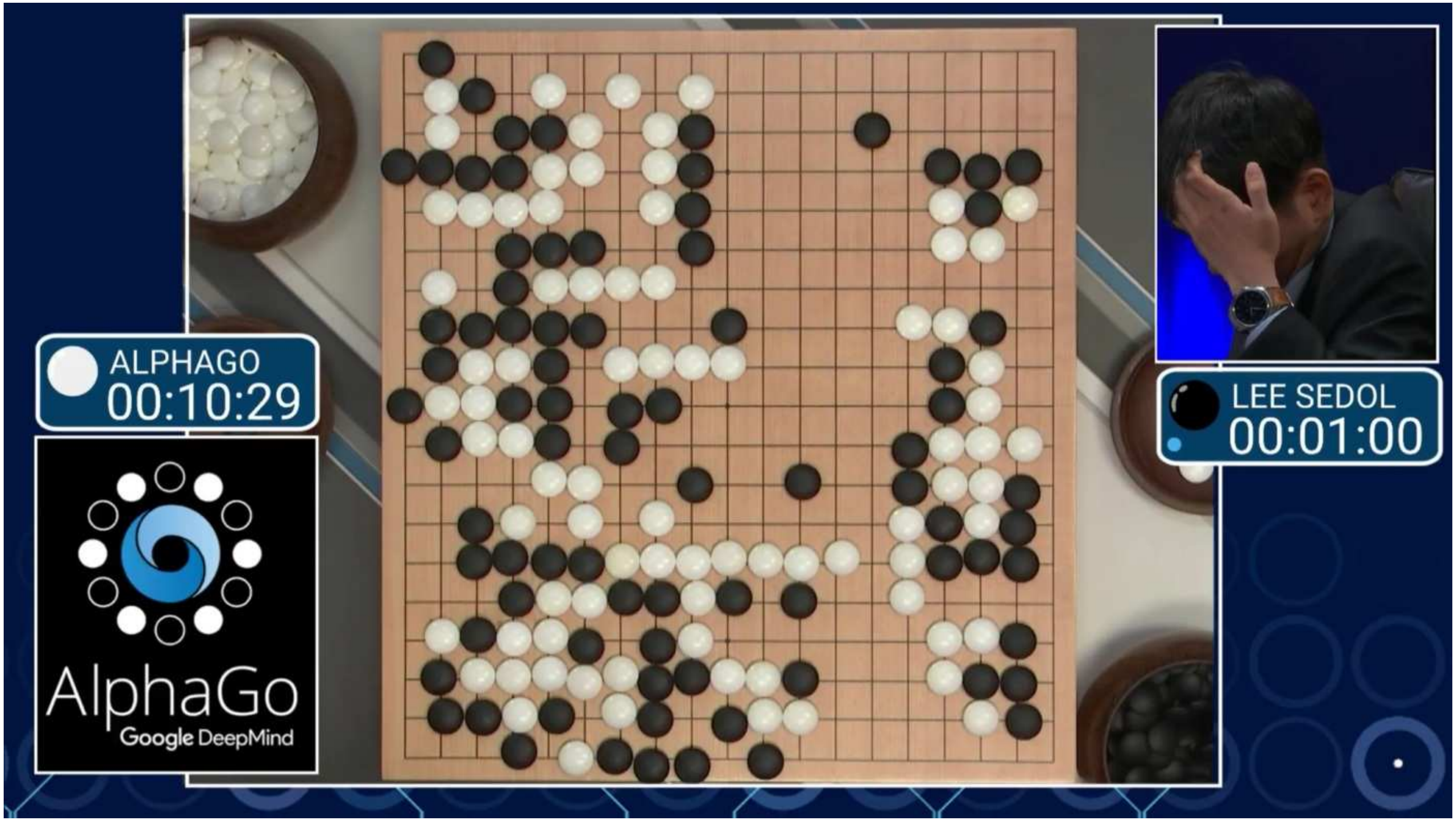
1 handout: slides

EOLQs

Are We Done?



Beyond A*

Greedy Best-first Search (GBFS)

$Q \leftarrow$ an ordered list containing just the initial state.
Loop
    If Q is empty,
        then return failure.
    Node $\leftarrow$ Pop(Q).
    If *Node* is a goal,
        then return *Node* (or path to it)
        else
            Children $\leftarrow$ Expand (*Node*).
            Merge *Children* into Q, keeping **sorted by heuristic**.$\leftarrow$

$h(n)$ = number of tiles out of place. (The blank is not a tile.)

Start state:

2	8	3
1	6	4
7	␣	5

Goal state:

1	2	3
8	␣	4
7	6	5

Please draw the tree resulting from the first two node expansions.

Evaluating Greedy

Assume branching factor b and solution at depth d.

Completeness:
Time:
Space:
Admissibility:

Beam Search

Truncate queue to hold the most promising k nodes.

k is the *beam width*.

Works best with breadth-first search!

Suboptimal Search

Problem Settings

optimal: minimize solution cost
suffer all with $f(n) = g(n) + h(n) < f^*$

---

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

anytime: iteratively converge to optimal

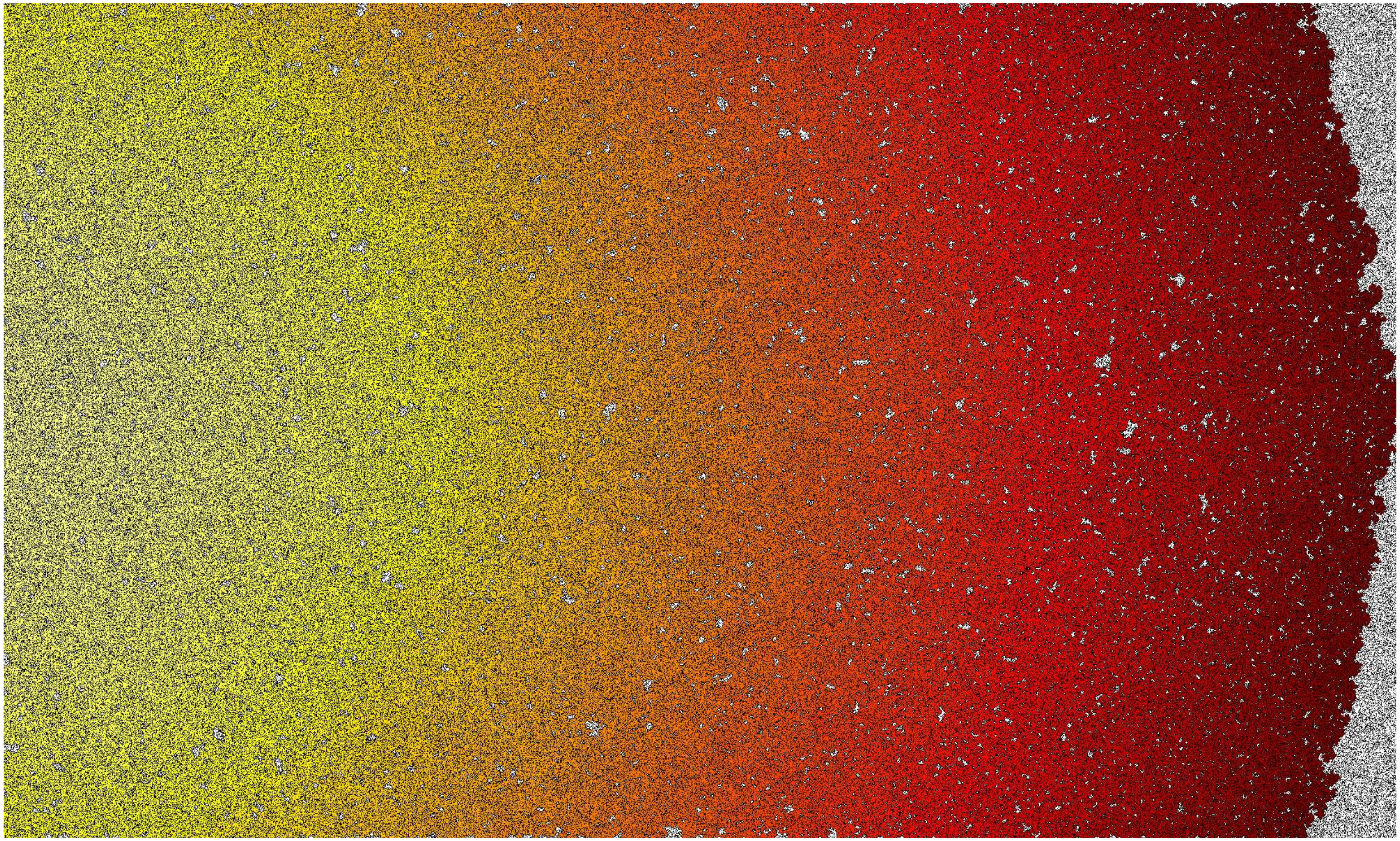
utility: maximize given function of cost and time

Weighted A*

$$f'(n) = g(n) + w \cdot h(n)$$

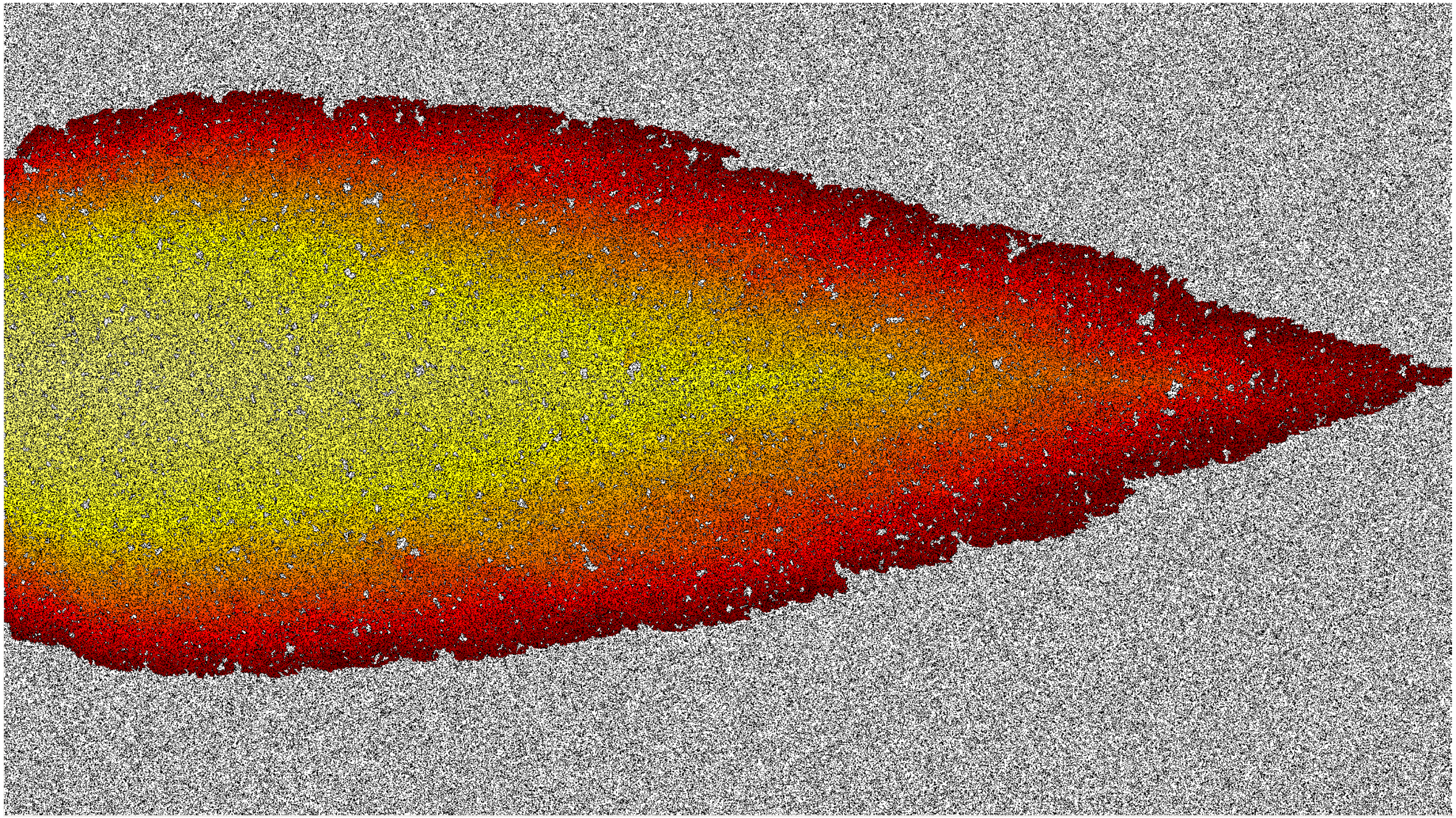
- nodes with high $h(n)$ look even worse
- no infinite rabbit holes
- suboptimality bounded: within a factor of w of optimal!

wA* Behavior



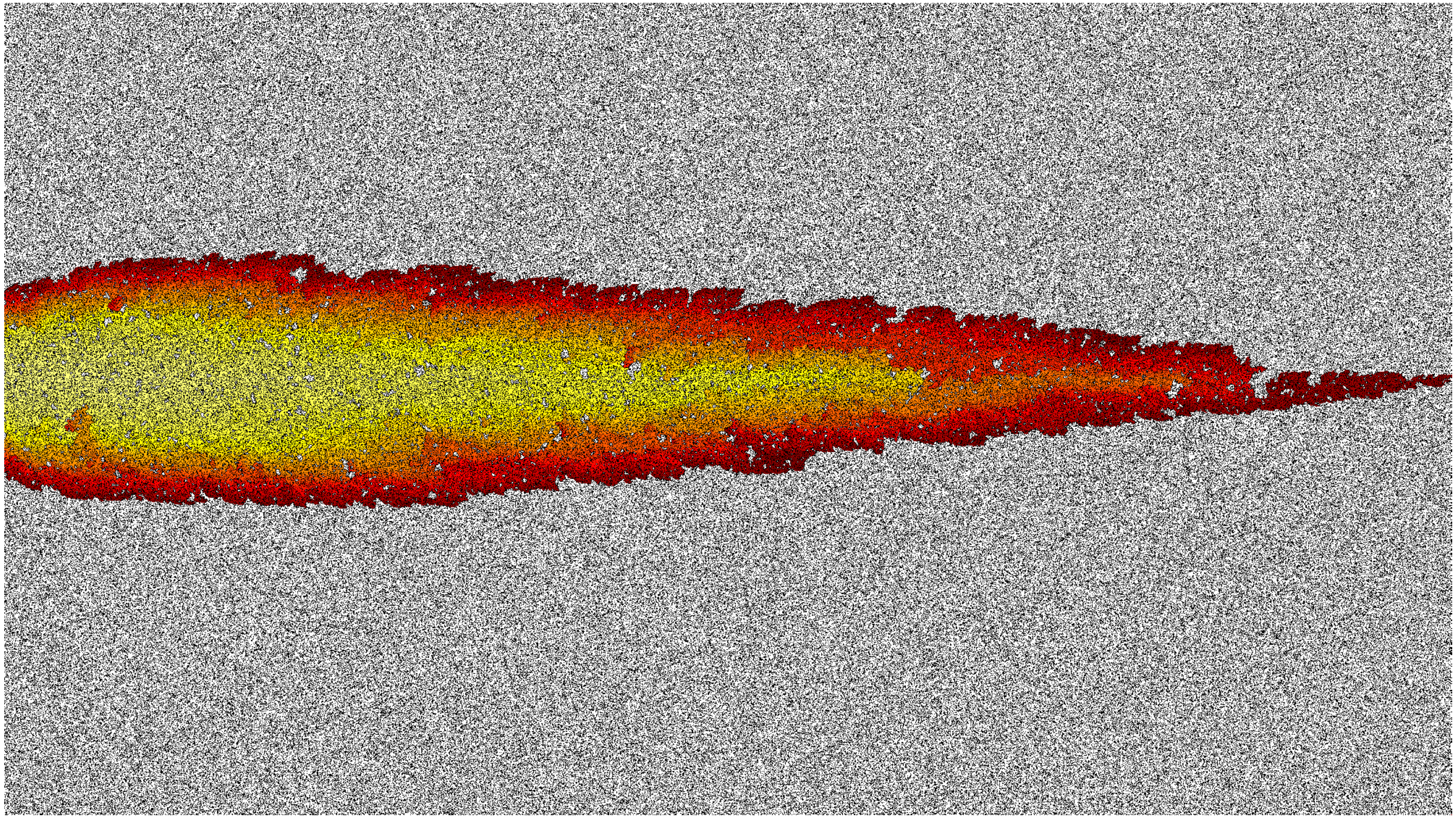
optimal: uniform-cost search

wA* Behavior



optimal: A*

wA* Behavior



bounded suboptimal: Weighted A*

For Speed: Distance-to-go, Not Cost-to-go

how to minimize solving time?

For Speed: Distance-to-go, Not Cost-to-go

how to minimize solving time?

how to minimize number of expansions?

For Speed: Distance-to-go, Not Cost-to-go

how to minimize solving time?

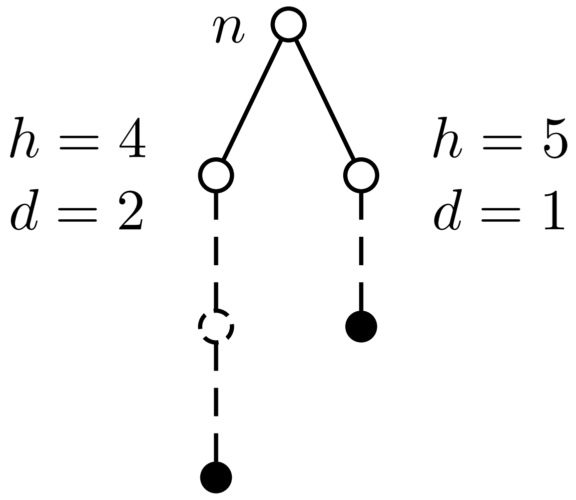
how to minimize number of expansions?

take the shortest path to a goal

For Speed: Distance-to-go, Not Cost-to-go

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 $h(n)$

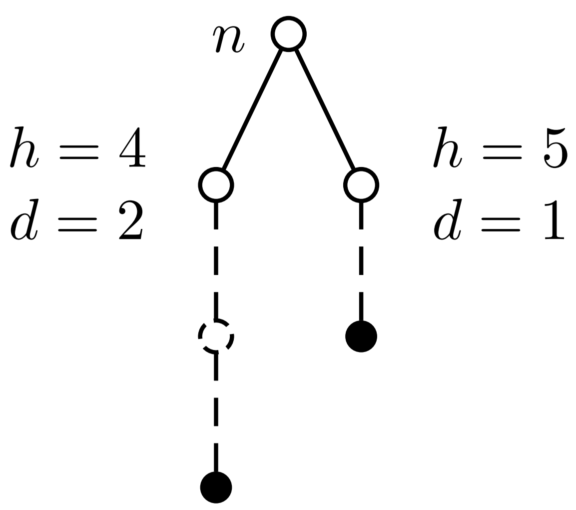
new information source: distance-to-go$= d(n)$



For Speed: Distance-to-go, Not Cost-to-go

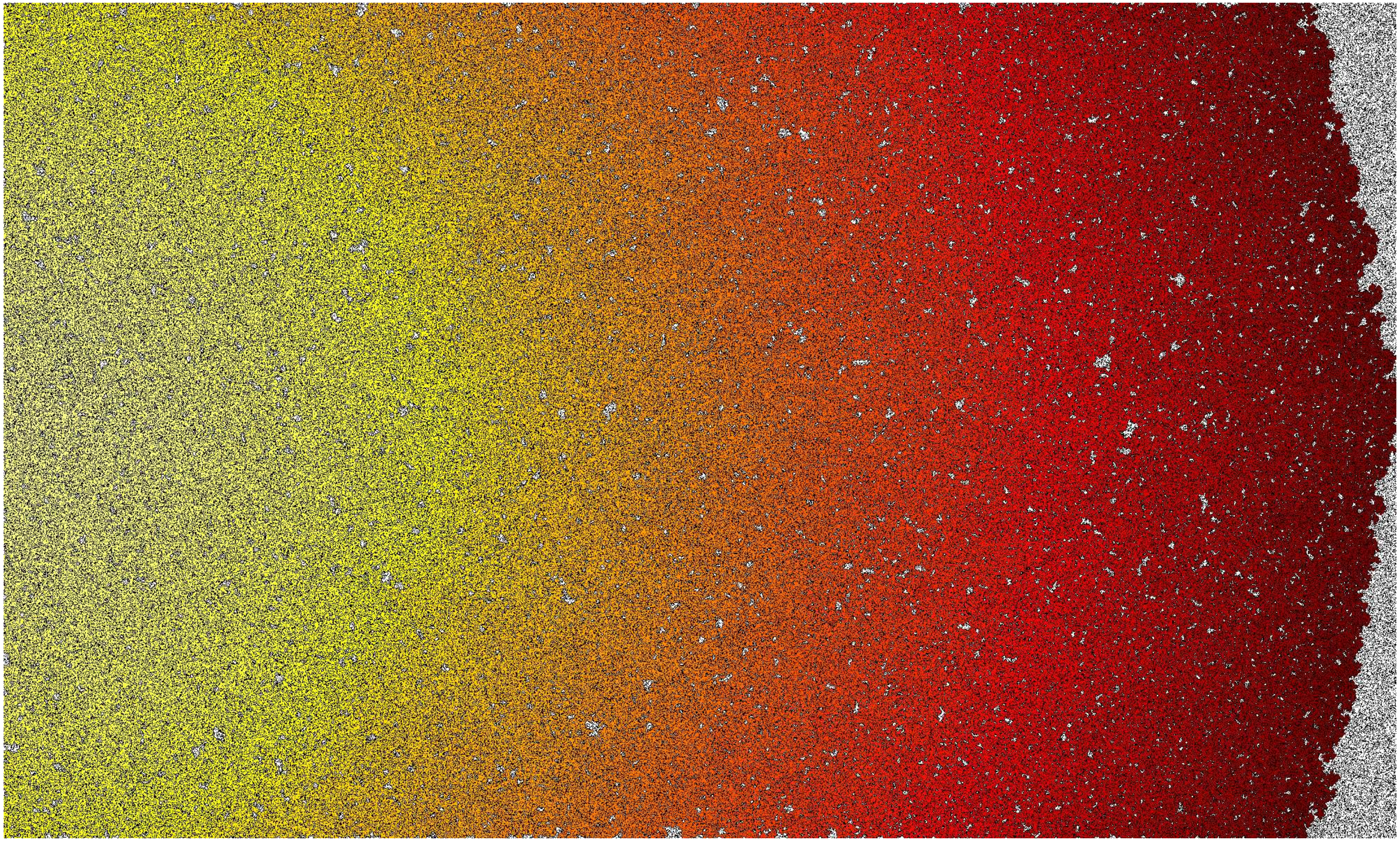
how to minimize solving time?
how to minimize number of expansions?
take the shortest path to a goal
for domains with costs, this is not $h(n)$

new information source: distance-to-go$= d(n)$



Speedy: best-first search on d

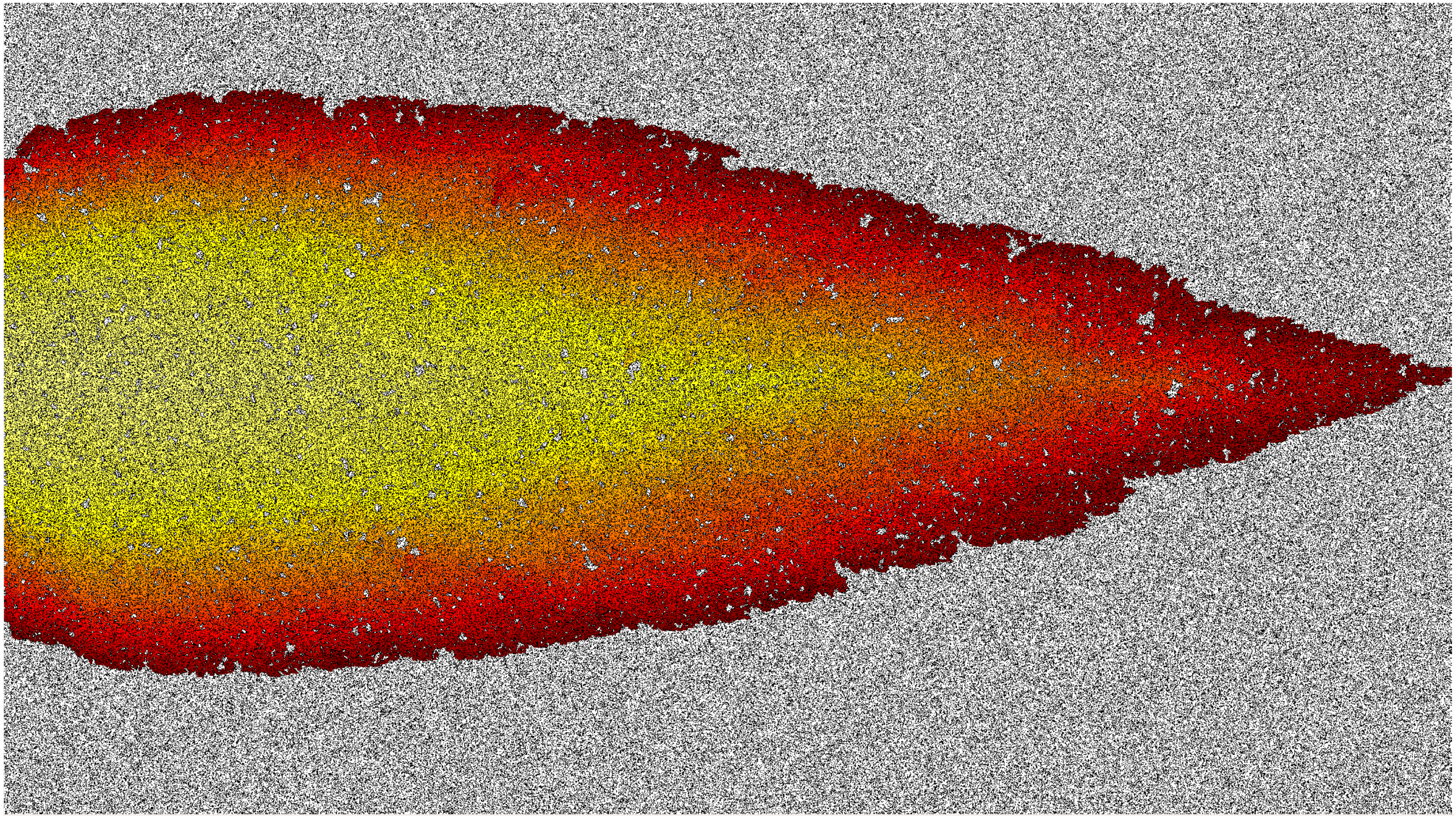
bounded-suboptimal using h, $\widehat{h}$, d



optimal: uniform-cost

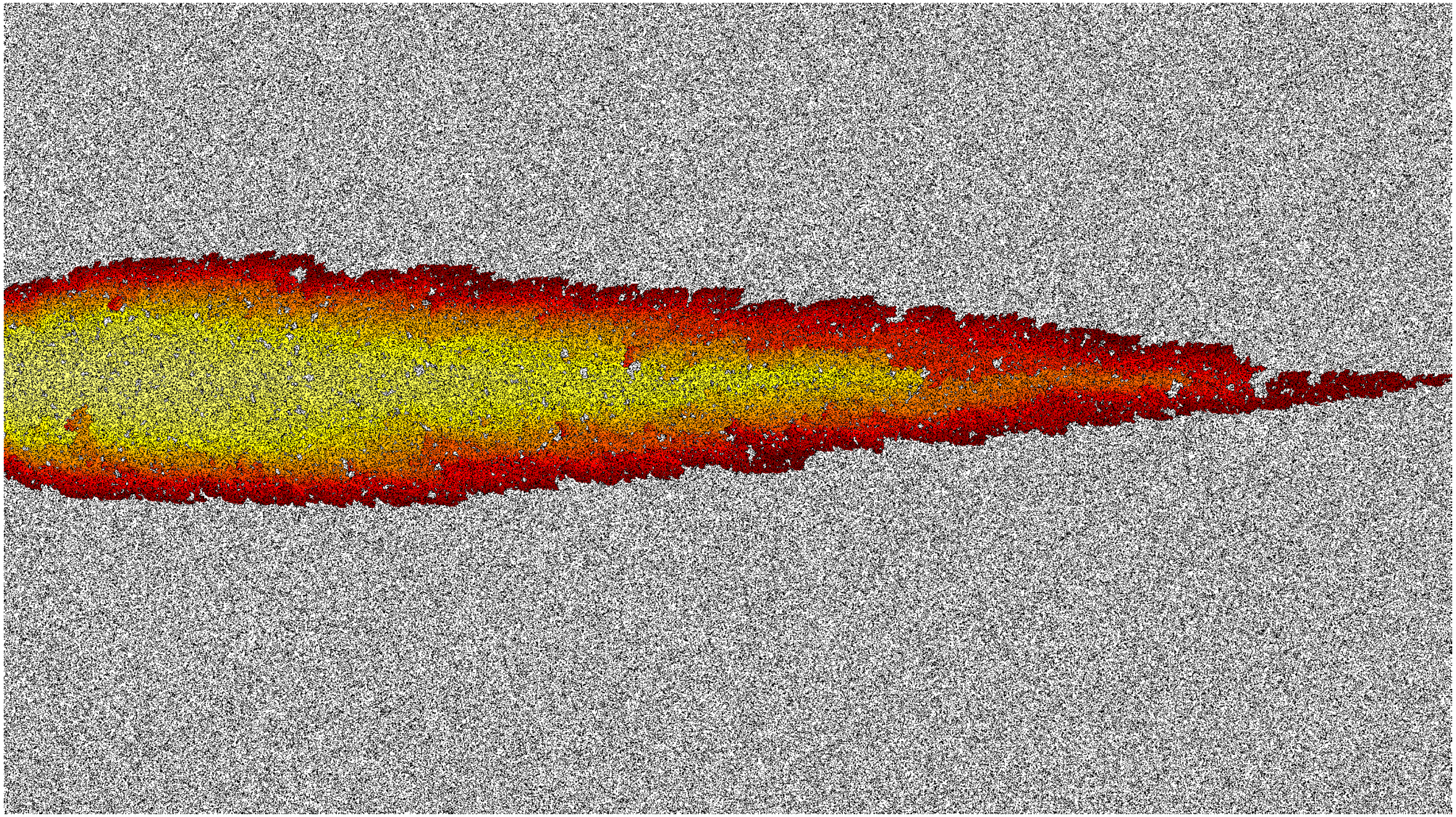
Round Robin d

bounded-suboptimal using h, $\widehat{h}$, d



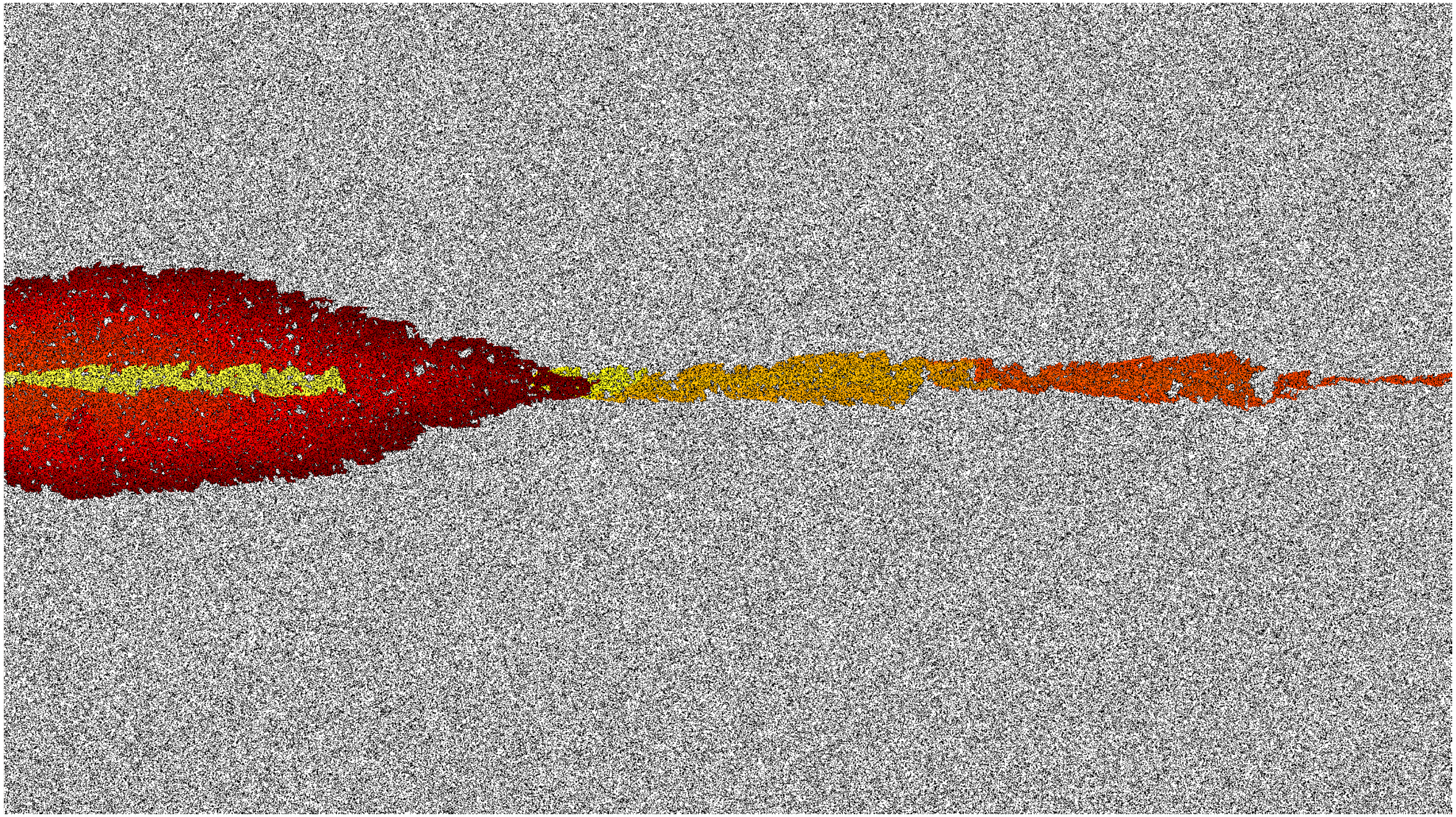
optimal: A*

bounded-suboptimal using h, $\widehat{h}$, d



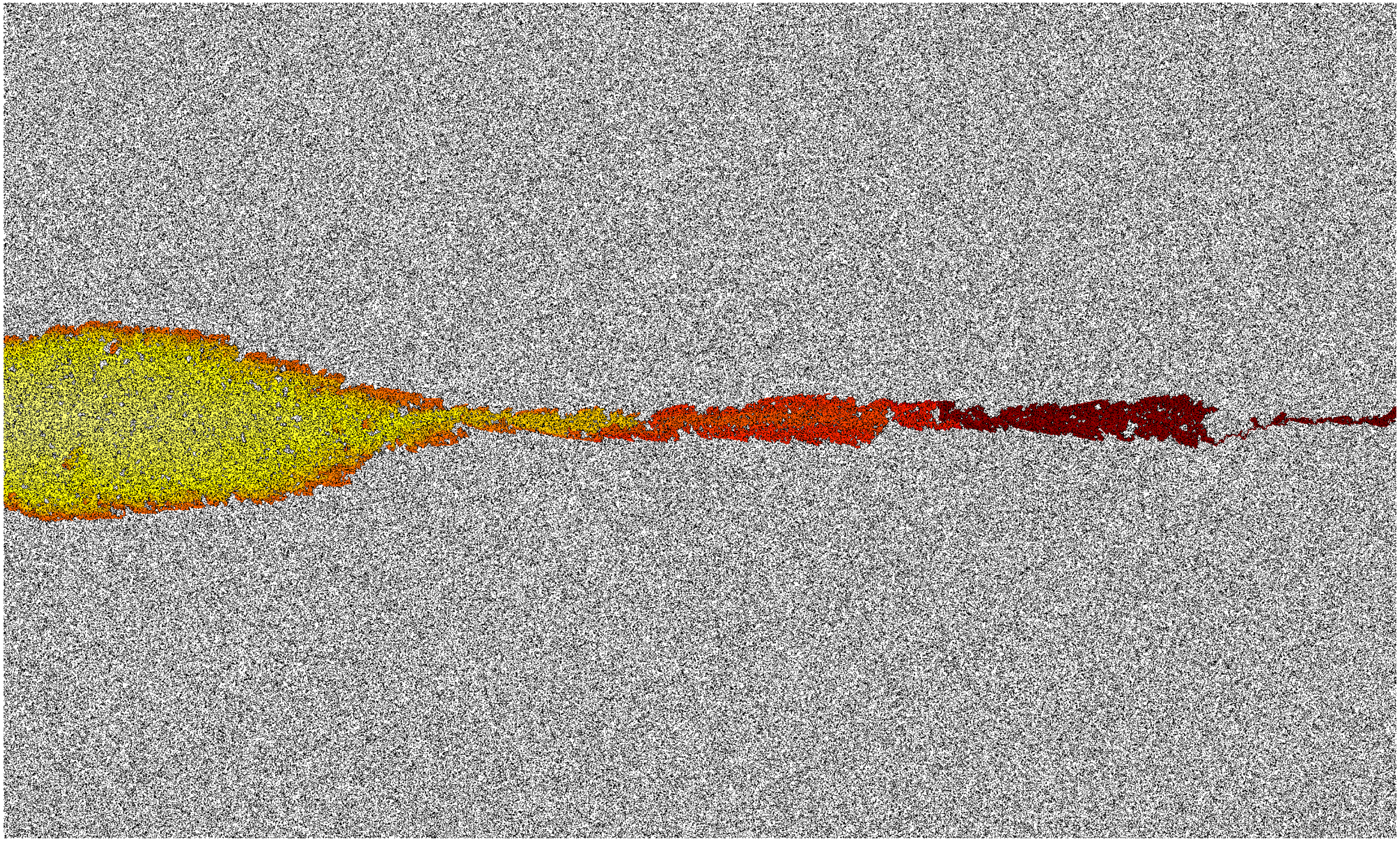
bounded suboptimal: Weighted A*

bounded-suboptimal using h, $\widehat{h}$, d



bounded suboptimal: Optimistic Search (ICAPS, 2008)

bounded-suboptimal using h, $\widehat{h}$, d



bounded suboptimal: Explicit Estimation Search (IJCAI, 2011)

Anytime Search

Anytime A*

1. run weighted A*
2. keep going after finding a goal
3. keep best goal found (can test at generation)
4. prune anything with $f(n) > incumbent$

Anytime Restarting A* (ARA*): lower weight after finding each solution
Anytime EES

Break

- asst2 (asst8), asst 3
- scores and grades

Real-time Search

RTA*

keep hash table of h values for visited states

1. for each neighbor of current state s
2. either find h in table or do some lookahead
3. add edge cost to get f
4. update $h(s)$ to second-best f value
5. move to best neighbor

LSS-LRTA*

1. single A* lookahead (LSS)
2. update all h values in LSS
3. move to frontier

Search Algorithms

Uninformed: DFS, UCS

Admissible: A*

Limited memory: iterative deepening (IDDFS, IDA*)

Satisficing: GBFS, Speedy, Beam

Bounded suboptimal: wA*, RR-d

Real-time: RTA*, LSS-LRTA*

Other Shortest-path Algorithms

- SMA*, IE
- RBFS
- Bugsy
- Rectangle Search
- any-angle pathfinding, Euclidean pathfinding
- multiobjective search
- multi-level planning: TAMP, MAPF

Course projects!

EOLQs

EOLQs

Please write down the most pressing question you have about the course material covered so far and put it in the box on your way out.

Thanks!