Search review

Anthony Gitter

gitter@biostat.wisc.edu

University of Wisconsin-Madison

Uninformed search

Overview of uninformed search algorithms

b: branching factor (assume finite) d: goal depth m: graph depth

	Complete	optimal	time	space
Breadth-first search	Y	Y, if [1]	$O(b^d)$	$O(b^d)$
Uniform-cost search[2]	Y	Y	$O(b^{C^*/\varepsilon})$	$O(b^{C^*/\varepsilon})$
Depth-first search	N	N	$O(b^m)$	$O(bm)$
Iterative deepening	Y	Y, if [1]	$O(b^d)$	$O(bd)$
Bidirectional search[3]	Y	Y, if [1]	$O(b^{d/2})$	$O(b^{d/2})$

1. edge costs constant
2. edge costs $\geq \varepsilon > 0$. C* is the best goal path cost.
3. both directions BFS; not always feasible.

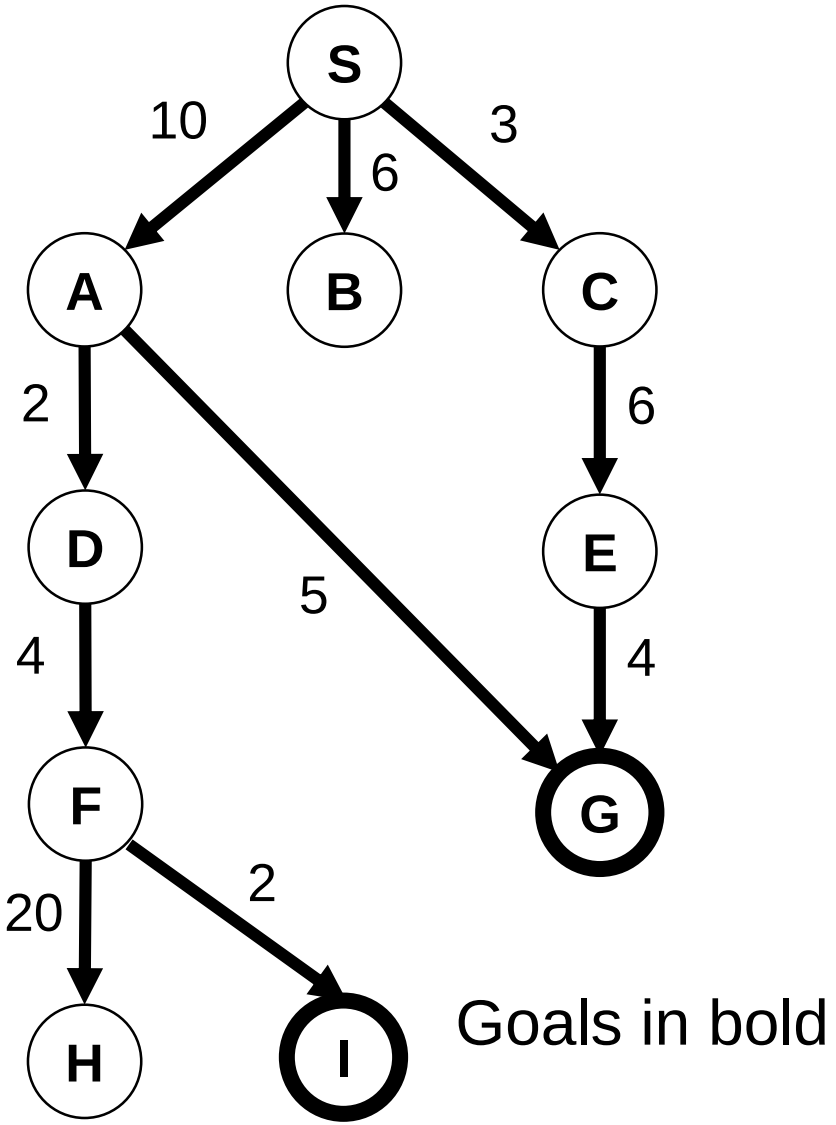
Unified view of uninformed search

- General state-space search algorithm with different node scores can implement various behaviors

- Breadth-first search
  - node score = node depth (queue behavior)
- Depth-first search
  - node score = -1 * node depth (stack behavior)
- Uniform-cost search
  - node score = path cost $g(s)$

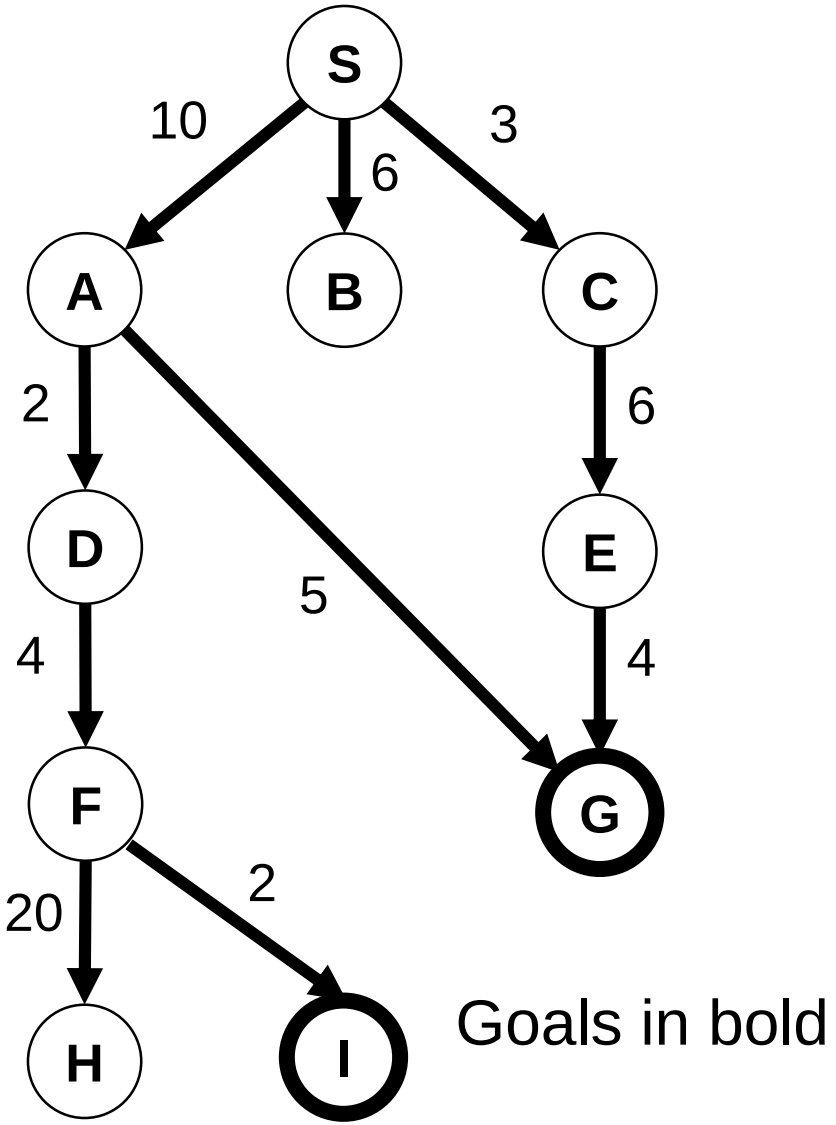
General search: breadth-first

Current	Priority queue (OPEN)
-	S(0)
S	A(1) B(1) C(1)
A	B(1) C(1) D(2) G(2)
B	C(1) D(2) G(2)
C	D(2) G(2) E(2)
D	G(2) E(2) F(3)
G	E(2) F(3)



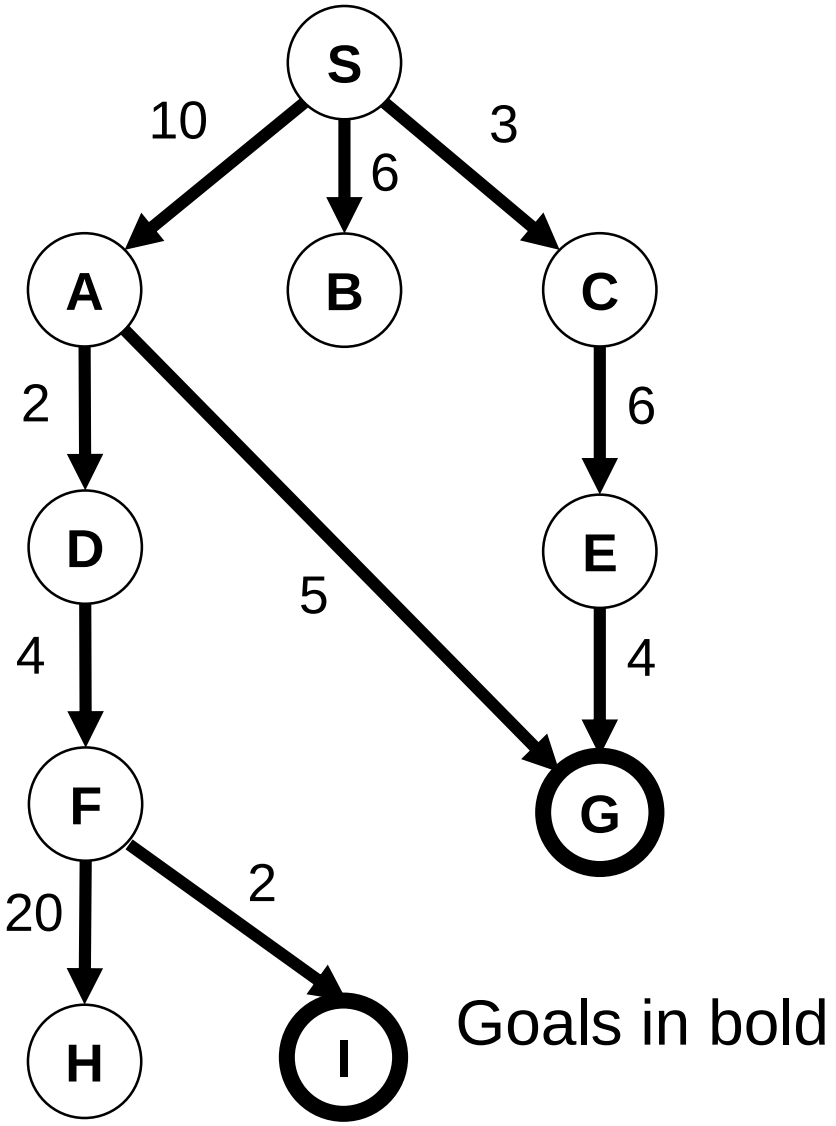
General search: depth-first

Current	Priority queue (OPEN)
-	S(0)
S	A(-1) B(-1) C(-1)
A	D(-2) G(-2) B(-1) C(-1)
D	F(-3) G(-2) B(-1) C(-1)
F	H(-4) I(-4) G(-2) B(-1) C(-1)
H	I(-4) G(-2) B(-1) C(-1)
I	G(-2) B(-1) C(-1)



General search: uniform-cost

Current	Priority queue (OPEN)
-	S(0)
S	C(3) B(6) A(10)
C	B(6) E(9) A(10)
B	E(9) A(10)
E	A(10) G(13)
A	D(12) G(13)
D	G(13) F(16)
G	F(16)



Informed search

Heuristics

- **Heuristic**: *h*(*s*), non-negative score that estimates remaining cost to goal
  - Difference between uninformed and informed
- **Valid heuristic**: Proposed node score that meets the definition of a heuristic
  - Always non-negative
- **Admissible heuristic**: valid heuristic that never over-estimates true cost to goal
  - Even for the goal state

Effects of good heuristics

- A good heuristic helps search avoid wasting time in unpromising regions of the state space

  Nodes generated in 8-puzzle with solution length *d*

d	**BFS**	**A*(h_1)**	**A*(h_2)**
6	128	24	19
8	368	48	31
10	1,033	116	48
20	91,493	9,905	1,318
28	463,234	202,565	22,055

h_1 is number misplaced tiles
h_2 is Manhattan distance

Data from Figure 3.26 Russell and Norvig 2020

Overview of informed search algorithms

	Node score	Heuristic	Optimal	Space savings
Uniform-cost search[1]	$g(s)$	None	Yes	None
Best-first greedy search	$h(s)$	Valid	No	None
A search	$g(s)+h(s)$	Valid	No	None
A* search	$g(s)+h(s)$	Admissible	Yes	None
IDA* search	$g(s)+h(s)$	Admissible	Yes	Does not expand nodes with score above threshold, iterative
Beam search	$g(s)+h(s)$	Admissible	No	Keep only k best nodes or nodes at most ε worse than the best

1. uninformed search, for comparison

Advanced search

(Un)informed search versus optimization

Uninformed and informed search	Optimization
Scores on states	Scores on states
Care about path from start to goal and path cost	Care about state with the best score
Successor function to move from one state to another	Generate neighbors (or cross-over + mutation)
Have learned several optimal algorithms	Focused on methods that may return local optimum
Many algorithms fail in massive state spaces	Have strategies for large or infinite state spaces

Neighborhoods in hill climbing

- Important to get right for local search
  - Ideally small neighborhood
  - Preserve structure in the problem
  - May require some ingenuity and domain knowledge
- What happens if the neighborhood is
  - Massive?
  - A random state?

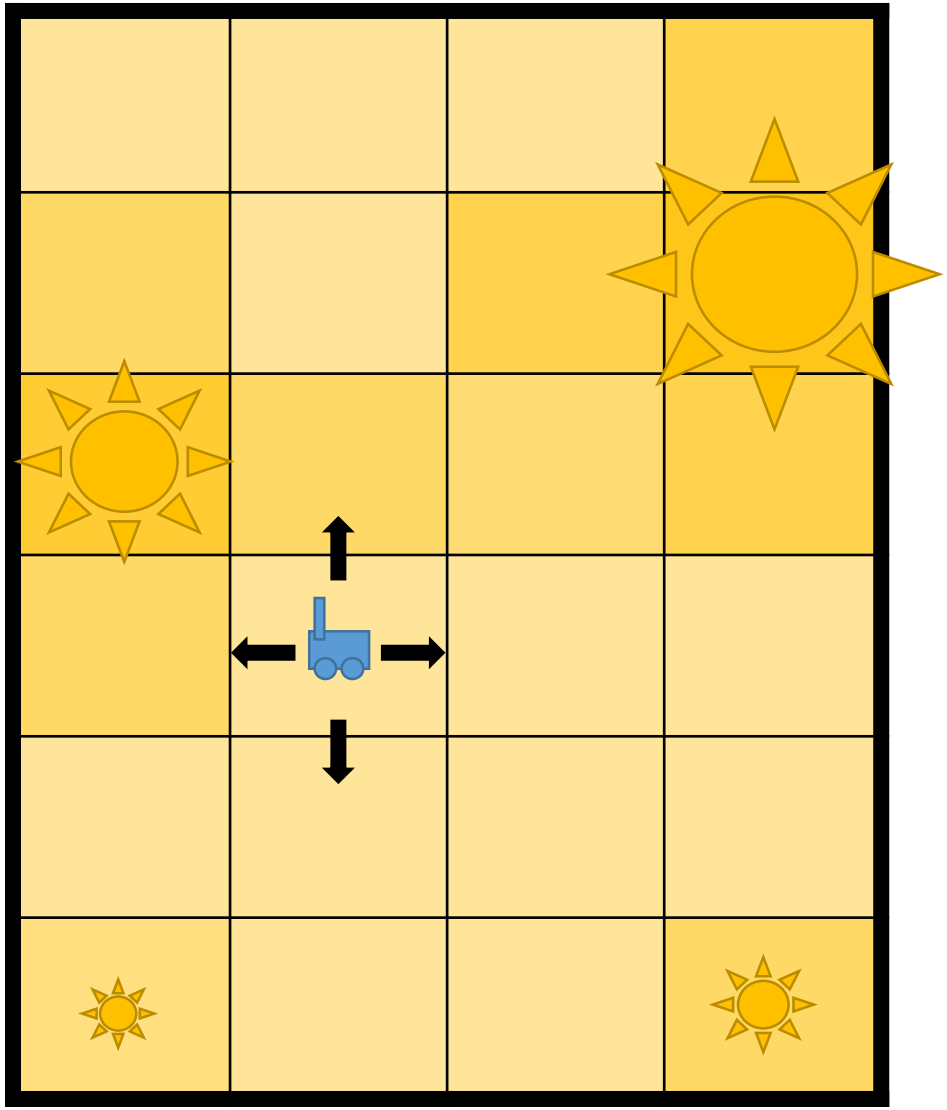
Comparing optimization algorithms

	Optimal	Susceptibility to getting stuck in local optima[1]	Greediness[1]	What neighbor to select?
Random search[2]	No	None	None	Does not use neighbors
Hill climbing	No	Extreme	Extreme	Best neighbor
Hill climbing with restarts	No	Some	Extreme	Best neighbor
Stochastic hill climbing[3]	No	High	High	Random better neighbor
First choice hill climbing[3]	No	High	High	First better neighbor
Simulated annealing	No	Some	Moderate	Better, or worse with small probability
Genetic algorithm	No	Some	Low	Generates “neighbor” population

1. hard to quantify, rough estimates 2. run for finite time 3. without restarts

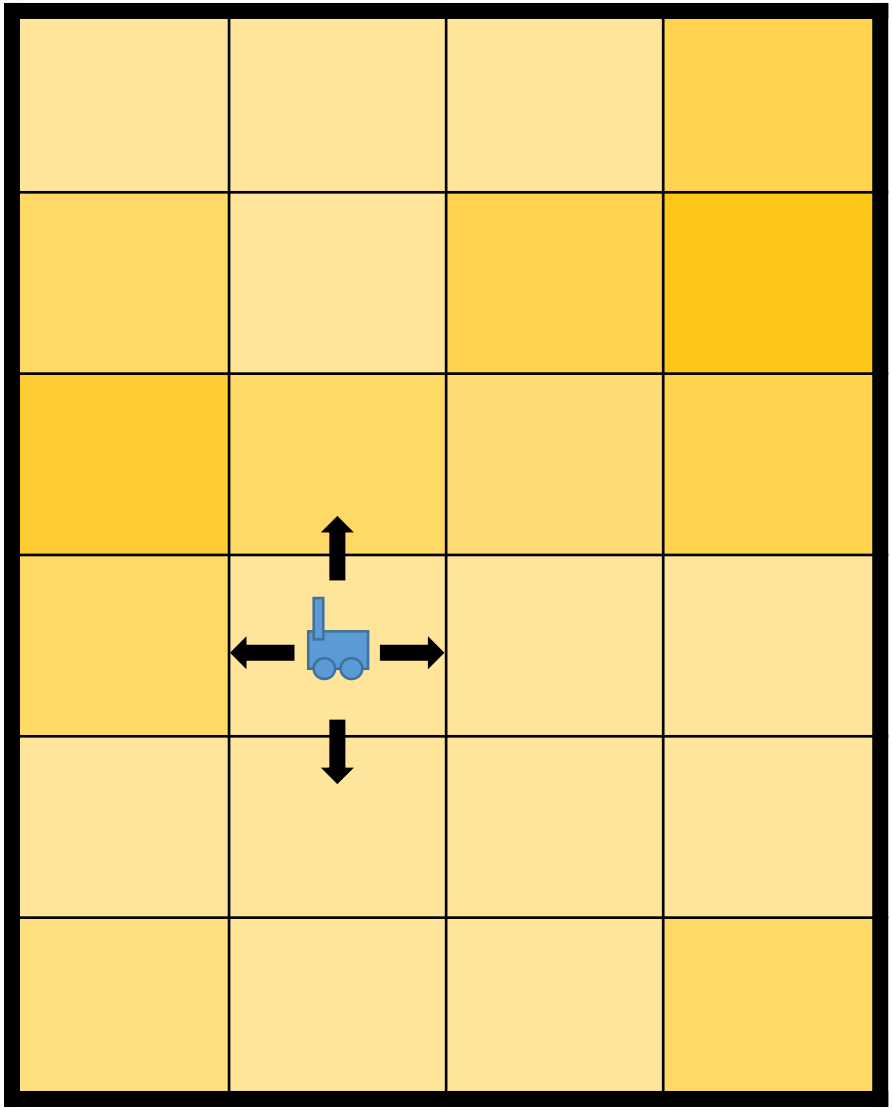
Simulated annealing example

- A robot in a large 2D room needs to find the location with the most sunlight so it can recharge.
- How does simulated annealing help?



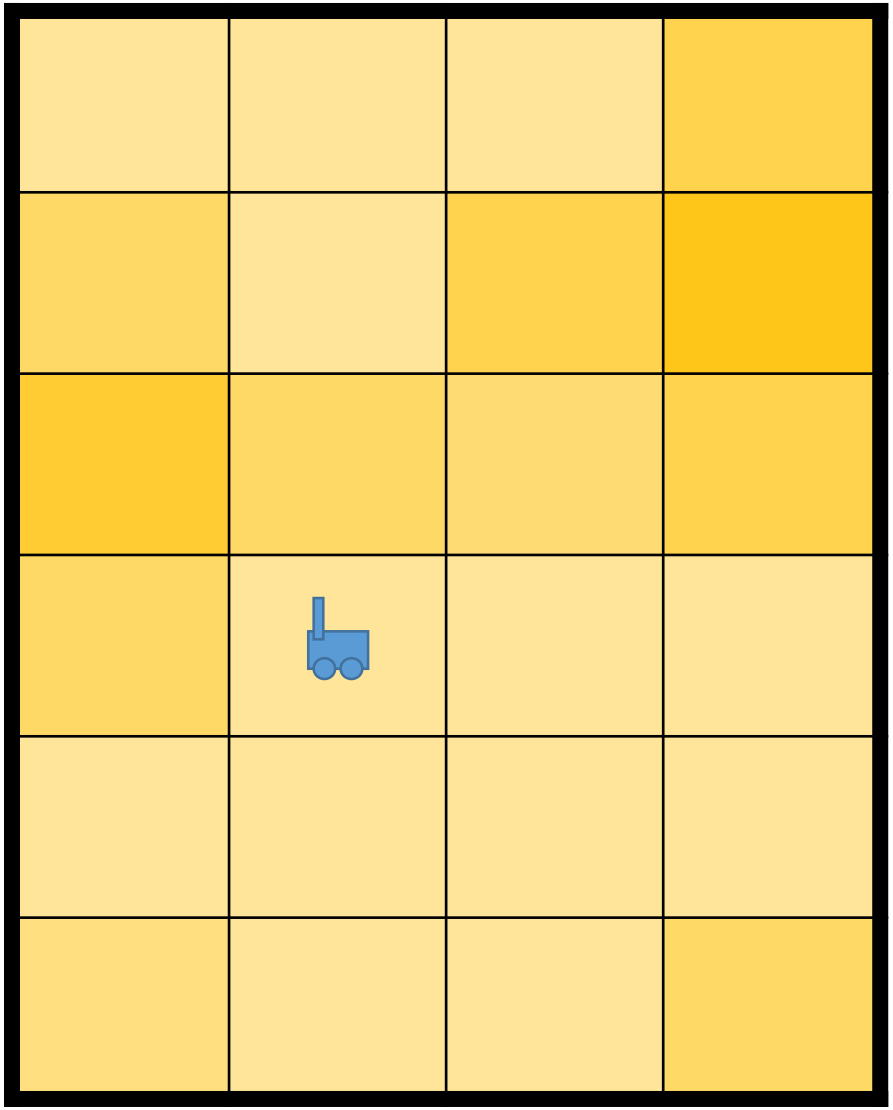
Simulated annealing example

- A robot in a large 2D room needs to find the location with the most sunlight so it can recharge.
- How does simulated annealing help?
- Darker positions have higher score



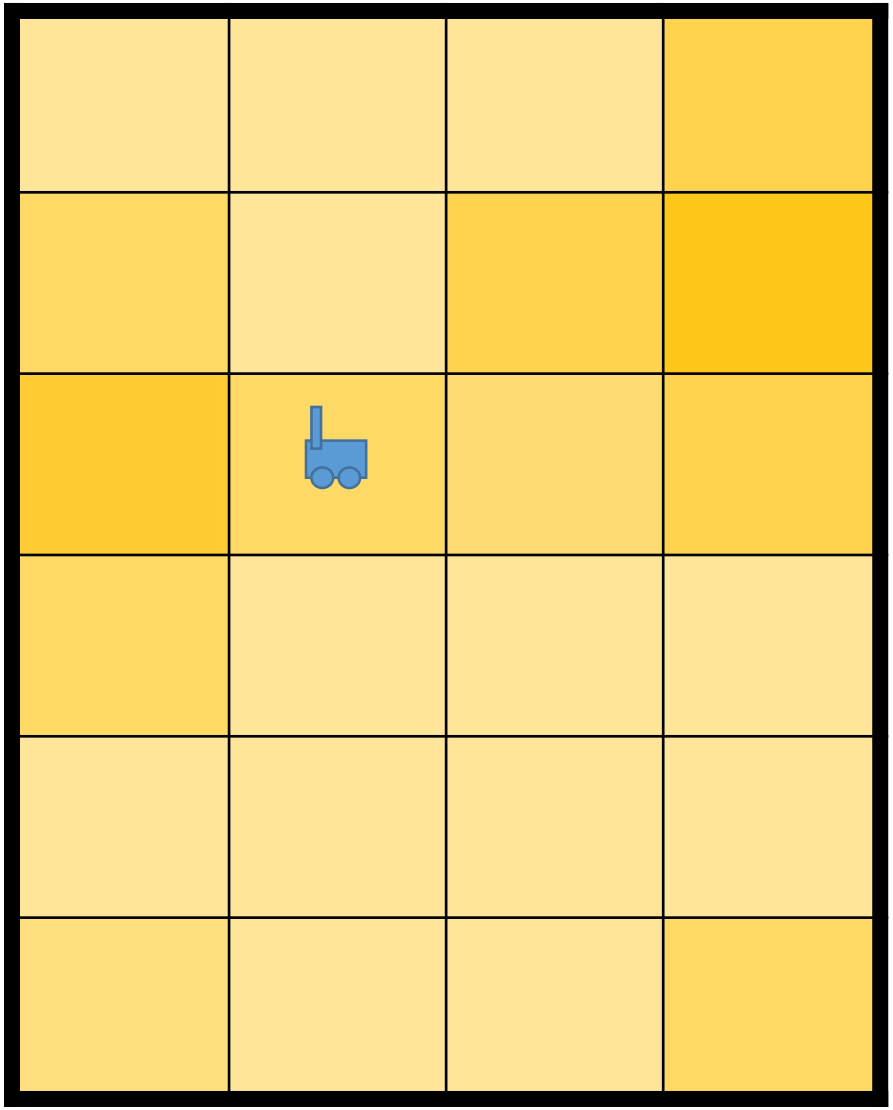
Simulated annealing example

- Hill climbing



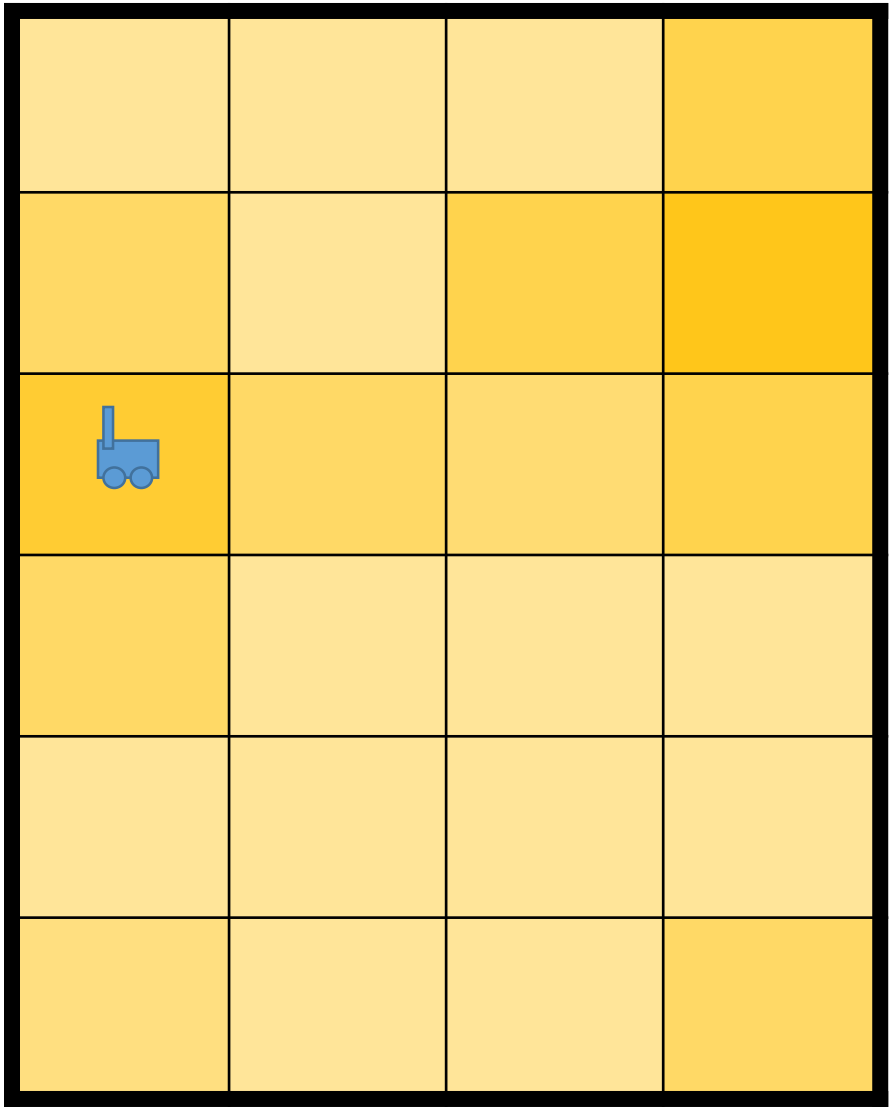
Simulated annealing example

- Hill climbing



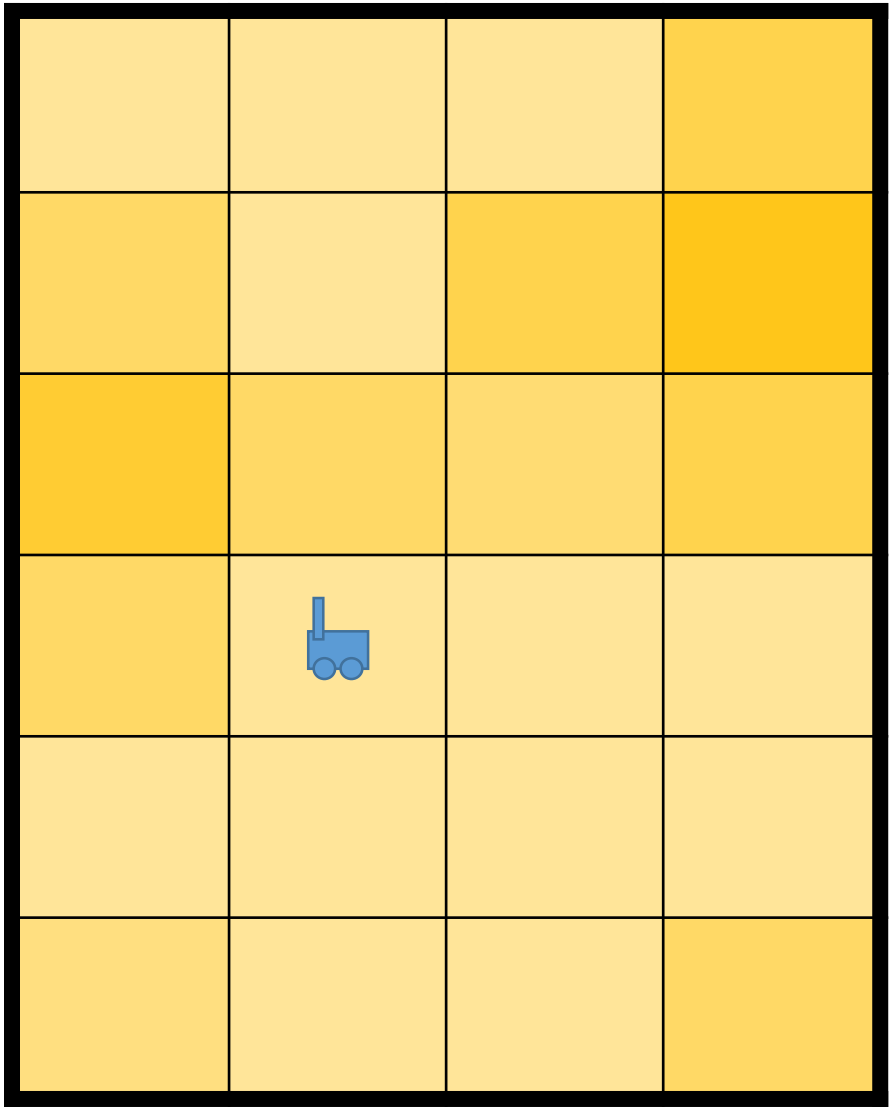
Simulated annealing example

- Hill climbing
- No better neighbor
- Stop



Simulated annealing example

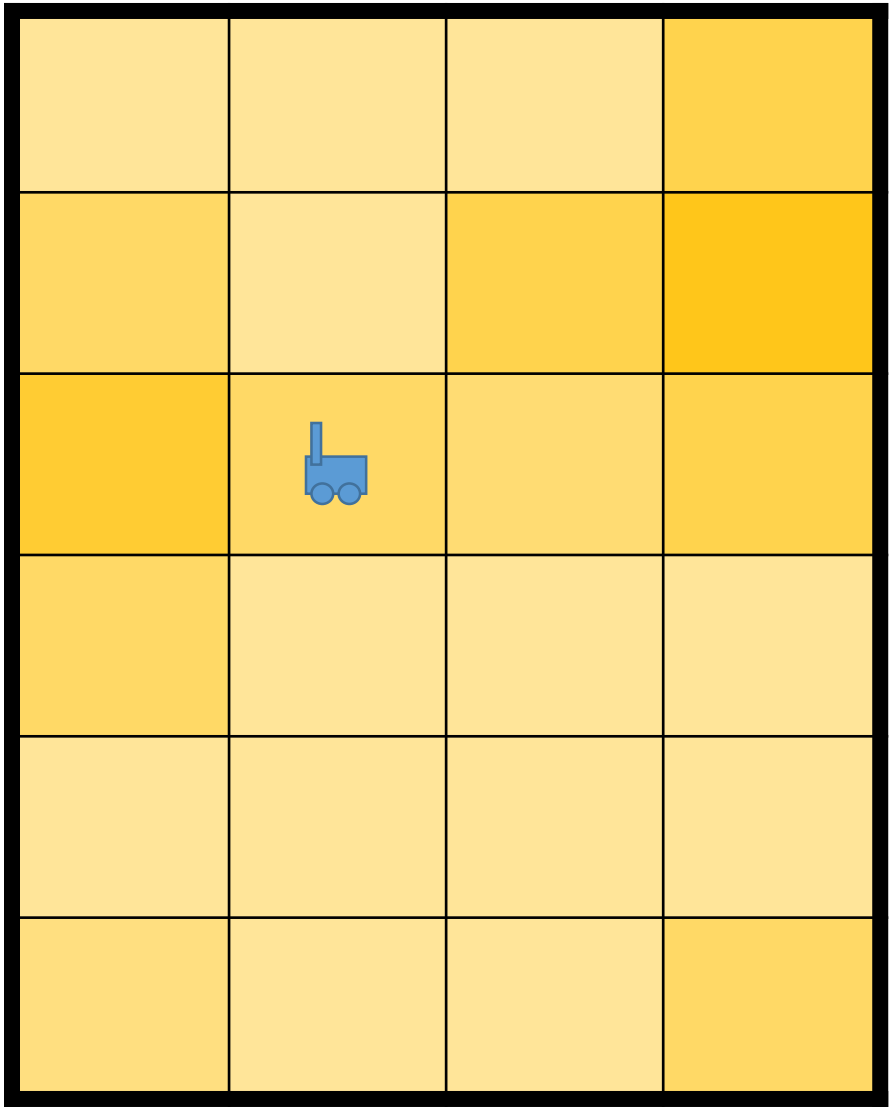
- Simulated annealing



Simulated annealing example

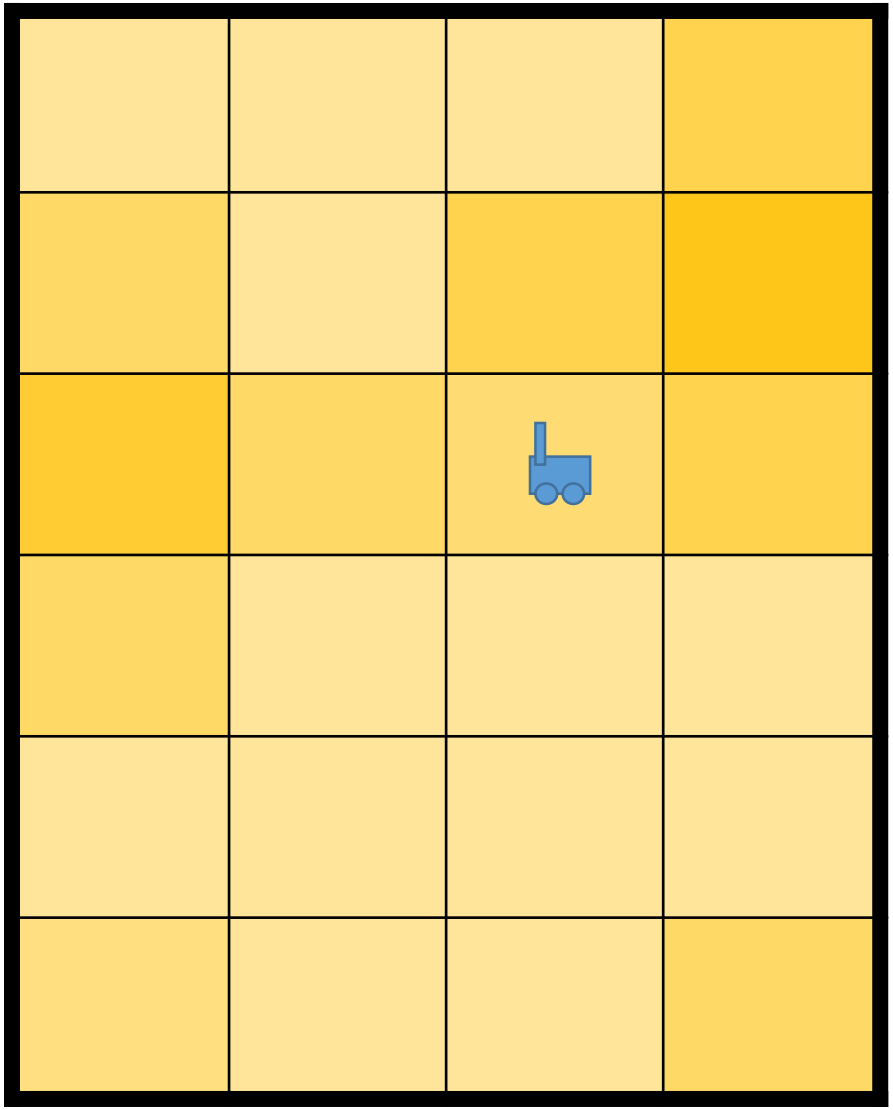
- Simulated annealing
- Only LEFT is better
- UP, DOWN, RIGHT all have worse scores

- However, select RIGHT with small probability



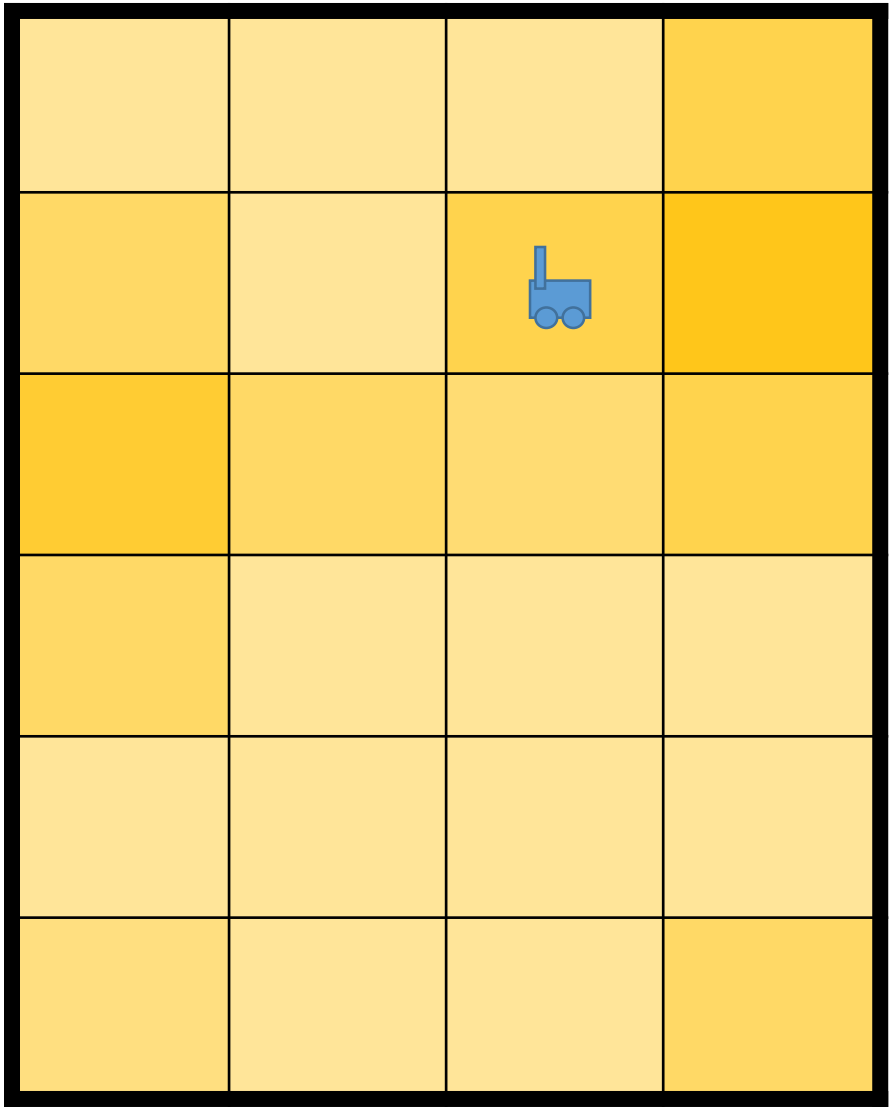
Simulated annealing example

- Simulated annealing
- Now UP, LEFT, and RIGHT all have better scores



Simulated annealing example

- Simulated annealing



Simulated annealing example

- Simulated annealing
- If the temperature schedule was good, robot is very unlikely to leave this global optimum