Heuristic Search

Alice Gao

Lecture 4

Readings: RN 3.5 (esp. 3.5.2), PM 3.6

Outline

Learning goals

By the end of the lecture, you should be able to

- Describe motivations for applying heuristic search algorithms.
- Trace the execution of and implement the Greedy best-first search and A* search algorithm with a given heuristic function.
- Describe properties of the Greedy best-first and A* search.
- Design an admissible heuristic function for a search problem.
- Describe reasons for choosing one heuristic function over another.

Why Heuristic Search?

How would ____ choose which one of the two states to expand?

- an uninformed search algorithm
- humans

5	3	
8	7	6
2	4	1

1	2	3
4	5	
7	8	6

Why Heuristic Search

An uninformed search algorithm

- considers every state to be the same.
- does not know which state is closer to the goal.
- may not find the optimal solution.

An heuristic search algorithm

using domain knowledge.

- uses heuristics to estimate how close the state is to a goal.
- try to find the optimal solution. (also find a solution faster.)

The Heuristic Function

Definition (search heuristic)

best/
shortest

A **search heuristic** $h(n)$ is an estimate of the cost of the cheapest path from node n to a goal node.

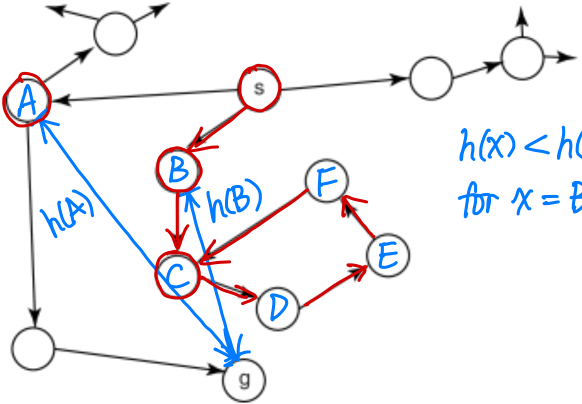
- $h(n)$ is arbitrary, non-negative, and problem-specific.
- If n is a goal node, $h(n) = 0$.
- $h(n)$ must be easy to compute (without search).

Greedy Best-First Search

- Frontier is a priority queue ordered by the heuristic $h(n)$.
- Expand the node with the lowest $h(n)$.

Greedy BFS: will it find a solution/terminate?

h: Euclidean straight line distance



$h(x) < h(A)$

for $x = B, C, D, E, F$

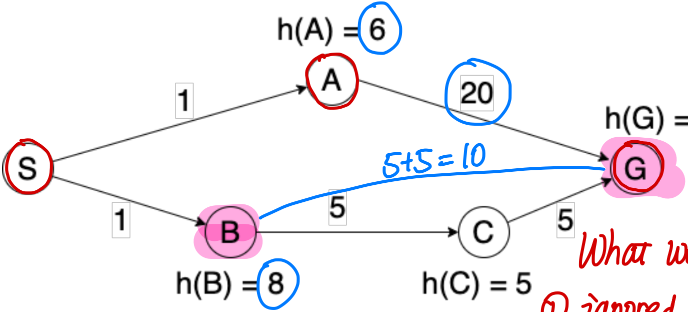
Greedy search gets stuck in the loop and never terminates.

Q: What about LCFS?

Greedy BFS: will it find the optimal solution?

frontier: ~~S~~, ~~SA~~, SB, ~~SAG~~ solution found: SAG

$h(A)=6$ $h(B)=8$ $h(G)=0$



What went wrong?

① ignored the costs.

② h values are deceiving.

optimal solution: SBCG

cost(SAG) = 21 cost(SBCG) = 11

Try Greedy Search on the grid search problem

- Number the nodes as they are removed from the frontier.
- Use multi-path pruning.
- Use Manhattan distance as our heuristic function.

tie breaking order : up, left, right, down.

A* Search

$$\underbrace{\underbrace{\text{start} \xrightarrow{\text{actual}} n}_{cost(n)} \underbrace{\xrightarrow{\text{estimate}} \text{goal}}_{h(n)}}_{f(n)}$$

- The frontier is a priority queue ordered by $cost(n) + h(n)$.
- Expand the node with the lowest $f(n)$.
- A mix of lowest-cost-first and greedy best-first search.
- Selects the node on the frontier with the lowest estimated distance from the start to a goal node constrained to go via that node.

Try A* Search on the grid search problem

- Number the nodes as they are removed from the frontier.
- Use multi-path pruning.
- Use Manhattan distance as our heuristic function.

tie breaking rule: up, left, right, down. remove most recently added node first.

f heuristic

order of expansion

cost

A* is Optimal

If the heuristic $h(n)$ is admissible,
the solution found by A* is optimal.

Definition (admissible heuristic)

A heuristic $h(n)$ is admissible if it is never an overestimate of the cost of the cheapest path from node n to a goal node.

$h^*(n)$ = the actual cost of the cheapest path from n to a goal node.

0

admissible.

$0 \leq h(n) \leq h^*(n)$

A* is Optimally Efficient

Among all optimal algorithms that start from the same start node and use the same heuristic, A* expands the fewest nodes.

Some Heuristic Functions for 8-Puzzle

- Manhattan Distance Heuristic: h_1
  The sum of the Manhattan distances of the tiles from their goal positions
- Misplaced Tile Heuristic: h_2
  The number of tiles that are NOT in their goal positions

Both heuristic functions are admissible.

Initial State (S)

$h_1(S) = 4+3+1+2+2+0+2+2 = 16$

5	3	
8	7	6
2	4	1

$h_2(S) = 7$

Goal State

1	2	3
4	5	6
7	8	

Constructing an Admissible Heuristic

- Define a relaxed problem by simplifying or removing constraints on the original problem.
- Solve the relaxed problem without search.
- The cost of the optimal solution to the relaxed problem is an admissible heuristic for the original problem.

Constructing an Admissible Heuristic for 8-Puzzle

8-puzzle: A tile can move from square A to square B

- if A and B are adjacent, and
- B is empty.

Which heuristics can we derive from relaxed versions of this problem?

CQ: Constructing an Admissible Heuristic

CQ: Which heuristics can we derive from the following relaxed 8-puzzle problem?

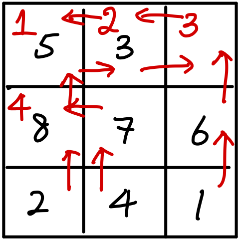
A tile can move from square A to square B
if A and B are adjacent. (B may not be empty.)

(A) The Manhattan distance heuristic

(B) The Misplaced tile heuristic

(C) Another heuristic not described above

$4+3+1+2$
$+ \cdots$



CQ: Constructing an Admissible Heuristic

CQ: Which heuristics can we derive from the following relaxed 8-puzzle problem? (A and B may not be adjacent.)

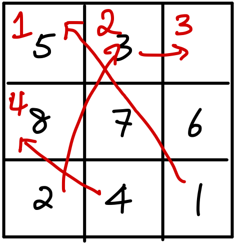
A tile can move from square A to square B. (B may not be empty.)

(A) The Manhattan distance heuristic

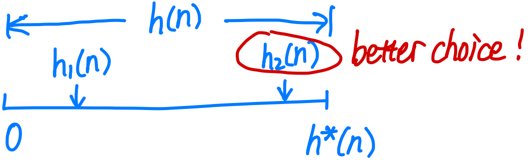
(B) The Misplaced tile heuristic

(C) Another heuristic not described above

1+1+1+1
+ .... +



Which Heuristic is Better?



- We want a heuristic to be admissible.
- Prefer a heuristic that is very different for different states.
- Want a heuristic to have higher values (close to h^*).

Dominating Heuristic

Definition (dominating heuristic)

Given heuristics $h_1(n)$ and $h_2(n)$. $h_2(n)$ dominates $h_1(n)$ if

- $(\forall n\ (h_2(n) \geq h_1(n)))$.
- $(\exists n\ (h_2(n) > h_1(n)))$.

Theorem

If $h_2(n)$ dominates $h_1(n)$, A using h_2 will never expand more nodes than A* using h_1.*

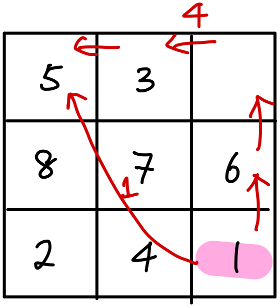
CQ: Which Heuristic of 8-puzzle is Better?

CQ: Which of the two heuristics of the 8-puzzle is better?

(A) The Manhattan distance heuristic dominates the Misplaced tile heuristic.

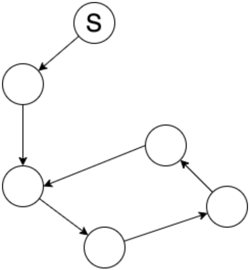
(B) The Misplaced tile heuristic dominates the Manhattan distance heuristic.

(C) I don't know....



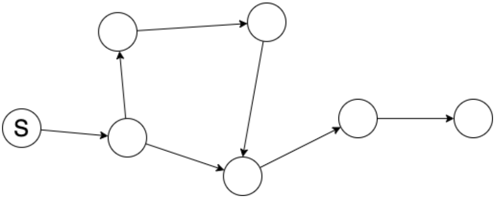
Cycle Pruning

- A cycle cannot be part of a least-cost path.
- Works well with depth-first search.
- The complexity of cycle pruning is . . .



Multiple-Path Pruning

- If we have already found a path to a node, we can prune other paths to the same node.
- Subsumes a cycle check.
- Requires storing all nodes we have found paths to.
- What if a subsequent path to *n* is shorter than the first path found?



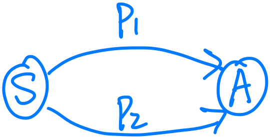
Lowest-cost-first search w/ multiple-path pruning

If we perform multiple-path pruning with lowest-cost-first search, is it possible for us to prune the optimal solution (least-cost path)?

(A) Yes, it is possible.

(B) No, it is not possible.

$\text{cost}(P_1) > \text{cost}(P_2)$



A* search w/ multiple-path pruning

Q: come up w/ a search graph s.t.
A* w/ multi-path pruning discards the optimal solution.

If we perform multiple-path pruning with A* search,
is it possible for us to prune the optimal solution (least-cost path)?

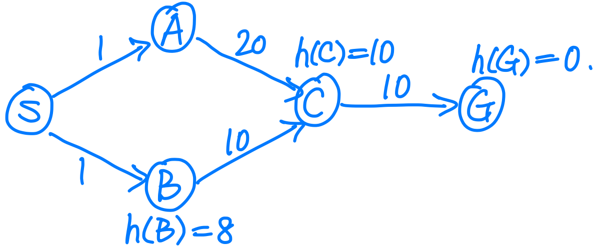
(A) Yes, it is possible.

(B) No, it is not possible.

frontier: ~~S~~, ~~SA~~, ~~SB~~, SAC, ~~SBC~~

f=7 f=9 f=31

h(A)=6



Find Optimal Solution w/ Multiple-Path Pruning

What if a subsequent path to n is shorter than the first path found?

- Remove all paths from the frontier that use the longer path.
- Change the initial segment of the paths on the frontier to use the shorter path.
- Make sure that the least-cost path to a node is found first.

A* search w/ multiple-path pruning

How can we ensure that A* with multiple-path pruning is optimal?

- Ensure that we find the least-cost path to every node first.
- Admissible heuristic guarantees the above for a goal node, but not for other nodes. → consistent.
- We need the heuristic to satisfy the monotone restriction:

$$\text{for any } arc\ \langle m, n \rangle,\ h(m) - h(n) \leq cost(m, n).$$

if n is a goal node, h(m) ≤ cost(m, n) → admissible heuristic.

If the heuristic satisfies the monotone restriction,
A* search with multiple-path pruning is optimal.

h(m) − h(n) = the heuristic estimate of the path cost from m to n.

Summary of Search Strategies

Strategy	Frontier Selection	Halts?	Space	Time
Depth-first	Last node added	No	Linear	Exp
Breadth-first	First node added	Yes	Exp	Exp
Lowest-cost-first	min $cost(n)$	Yes	Exp	Exp
Greedy Best-first	min $h(n)$	No	Exp	Exp
A*	min $cost(n) + h(n)$	Yes	Exp	Exp

Revisiting the learning goals

By the end of the lecture, you should be able to

- Describe motivations for applying heuristic search algorithms.
- Trace the execution of and implement the Greedy best-first search and A* search algorithm with a given heuristic function.
- Describe properties of the Greedy best-first and A* search.
- Design an admissible heuristic function for a search problem.
- Describe strategies for choosing among multiple heuristic functions.
- Describes strategies for pruning a search space.