Informed Search

Alice Gao
Lecture 4

Based on work by K. Leyton-Brown, K. Larson, and P. van Beek

Outline

Learning goals

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

- Define/trace/implement informed search algorithms (with/without cost) (handling cycles and repeated states).
- Determine properties of search algorithms: completeness, optimality, time and space complexity.
- Select the most appropriate search algorithms for specific problems.
- Construct admissible heuristics for appropriate problems. Verify heuristic dominance.

Properties of Uninformed Search Strategies

Algorithm	Complete?	Optimal?	Time	Space
IDS	Yes*	Yes***	$O(b^d)$	$O(bd)$
DFS	Yes**	No	$O(b^m)$	$O(bm)$
BFS	Yes*	Yes***	$O(b^d)$	$O(b^d)$

* if the branching factor is finite.

** if the graph is finite and does not contain cycles.

*** if all arc costs are the same.

Informed Search

Domain-specific knowledge

- can help people solve hard problems without search.
- can help computers find solutions more efficiently.

Informed/Heuristic search

- Estimate the cost from a given node to a goal node.
- Take into account of the goal when selecting the path to explore.

Our goal

- Our goal is to find the cheapest path from the start node to a goal node.
- $f^*(n)$:
- $f^*(n)$ is impossible to know. Thus, we estimate it.

Estimating the cost of the optimal path

$f(n)$:

Two functions we can use to construct $f(n)$:

- $g(n)$:
- $h(n)$:

The Heuristic Function

Definition (search heuristic)

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).

Three New Search Algorithms

Treat the frontier as a priority queue ordered by $f(n)$.
Expand the node with the lowest $f(n)$.
The choice of f determines the search strategy.

Uninformed search algorithm:

- Lowest-cost-first search: $f(n) = g(n)$.

Informed search algorithms (that use $h(n)$):

- Greedy search: $f(n) = h(n)$.
- A*: $f(n) = g(n) + h(n)$.

Quiz 1

Alas! Time for Quiz 1!

Good luck!

Lowest-Cost-First Search (Uniform-Cost Search)

- Goal: minimize the cost of the path to node n.
- Treat the frontier as a priority queue ordered by $f(n) = g(n)$.
- Expand the cheapest node

- Complete?
- Optimal?
- Time complexity: $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ where C^* is the cost of the optimal path and every arc cost exceeds $\epsilon > 0$.
- Space complexity: $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ where C^* is the cost of the optimal path and every arc cost exceeds $\epsilon > 0$.

CQ: Is Lowest-Cost-First Search Optimal?

CQ: Is Lowest-Cost-First Search optimal? Assume that every arc cost exceeds $\epsilon > 0$ and the branching factor *b* is finite.

(A) Yes

(B) No

(C) Not enough information to tell

Greedy Search (Best-First Search)

- Goal: minimize the estimated cost to the goal.
- Treat the frontier as a priority queue ordered by $f(n) = h(n)$.
- Try to get as close to the goal as it can.

- Complete?
- Optimal?
- Time complexity: $O(b^m)$
- Space complexity: $O(b^m)$

CQ: Is Greedy Search Complete?

CQ: Does there exist a search problem and a heuristic function such that Greedy Search is NOT complete on the problem?

(A) Yes

(B) No

CQ: Is Greedy Search Optimal?

CQ: Does there exist a search problem and a heuristic function such that Greedy Search is NOT optimal on the problem?

(A) Yes

(B) No

A* Search

- Goal: Minimize the estimated cost of the cheapest path from the start node to the goal through the current node n.
- $f(n) = g(n) + h(n)$
- Complete? Yes, if all arc costs exceed some $\epsilon > 0$ and b is finite.
- Optimal? Yes, if the heuristic is admissible, all arc costs exceed some $\epsilon > 0$, and b is finite.
- Time complexity: $O(b^m)$
- Space complexity: $O(b^m)$

A* is Optimal

The solution found by A* search is optimal if the heuristic $h(n)$ is admissible.

Definition (admissible heuristic)

A search heuristic $h(n)$ is admissible if it is never an overestimate of the cost from node n to a goal node. That is, $(\forall n\ (h(n) \leq h^*(n)))$.

- An admissible heuristic is a lower bound on the cost of getting from node n to the nearest goal node.

A* is Optimally Efficient

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

- No algorithm with the same information can do better.
- Intuition: any algorithm that does not expand all nodes with $f(n) < C^*$ run the risk of missing the optimal solution.

Comparing LCFS, GS and A*

Algorithm	Complete?	Optimal?	Time	Space
A*				
GS				
LCFS				

Iterative Deepening A*

- Each iteration is Depth-First Search until a f-value threshold.
- A node is not added to the frontier if its f value exceeds the threshold.
- Next iteration sets the new threshold to be the smallest f-value that exceeded the old threshold.

Examples of Heuristic Functions

8-Puzzle:

- The number of tiles out of place
- The sum of the Manhattan distances of the tiles from their goal positions

River Crossing:

- The number of objects that still need to get to the other side of the river.

CQ: Is this heuristic admissible?

CQ: Is the following heuristic for the river crossing problem admissible?

$h(n) =$ the number of objects that still need to get to the other side of the river.

(A) Yes

(B) No

(C) Not enough information to tell

Constructing an Admissible Heuristic

- Define a relaxed problem by simplifying or dropping requirements 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

Example: 8-puzzle: A tile can move from A to B
if A and B are adjacent and B is blank.

Which heuristics can we derive from the relaxed problems below?

- Relaxed problem 1: A tile can move from A to B
  if A and B are adjacent.
- Relaxed problem 2: A tile can move from A to B
  if B is blank.
- Relaxed problem 3: A tile can move from A to B.

CQ: Constructing an Admissible Heuristic

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

Relaxed problem 1: A tile can move from A to B if A and B are adjacent.

(A) The number of tiles out of place

(B) The sum of the Manhattan distances of the tiles from their goal positions

(C) Another heuristic not described above

CQ: Constructing an Admissible Heuristic

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

Relaxed problem 3: A tile can move from A to B.

(A) The number of tiles out of place

(B) The sum of the Manhattan distances of the tiles from their goal positions

(C) Another heuristic not described above

Which Heuristic is Better?

- We want a heuristic to be admissible.
- We don't want a heuristic to be close to a constant function.
- We 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.*

Revisiting the learning goals

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

- Define/trace/implement informed search algorithms (with/without cost) (handling cycles and repeated states).
- Determine properties of search algorithms: completeness, optimality, time and space complexity.
- Select the most appropriate search algorithms for specific problems.
- Construct admissible heuristics for appropriate problems. Verify heuristic dominance.