Foundations of Artificial Intelligence

CS472/3 — Fall 1999

Lecture #4

Bart Selman

Today's Lecture

Informed Search

Readings: R&N, Chapter 4.

Informed Methods: Heuristic Search

Informed Methods use problem-specific knowledge.

Heuristic search is an attempt to search the most promising paths first. Uses heuristics, or rules of thumb, to find the best node to expand next.

Relies on an *evaluation function* — indicates the desirability of expanding a node. E.g., *path cost*.

$h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state (*Heuristic Function*: e.g., straight line on a map)

General Principle

Given a list of nodes to be expanded, choose the one that the heuristic function estimates as the most promising.

Best-First Search

1. Set L to be the initial node(s).
2. Let n be the node on L that is "most promising" according to eval function ($h(n)$ / $f(n)$). If L is empty, fail.
3. If n is a goal node, stop and return it (and the path from the initial node to n).
4. Otherwise, remove n from L and add all of n's children to L (labeling each with its path from the initial node). Return to step 2.

Two Instantiations of Best-First Search

Issue: Best-First depends on *evaluation function*.

Alternatives:

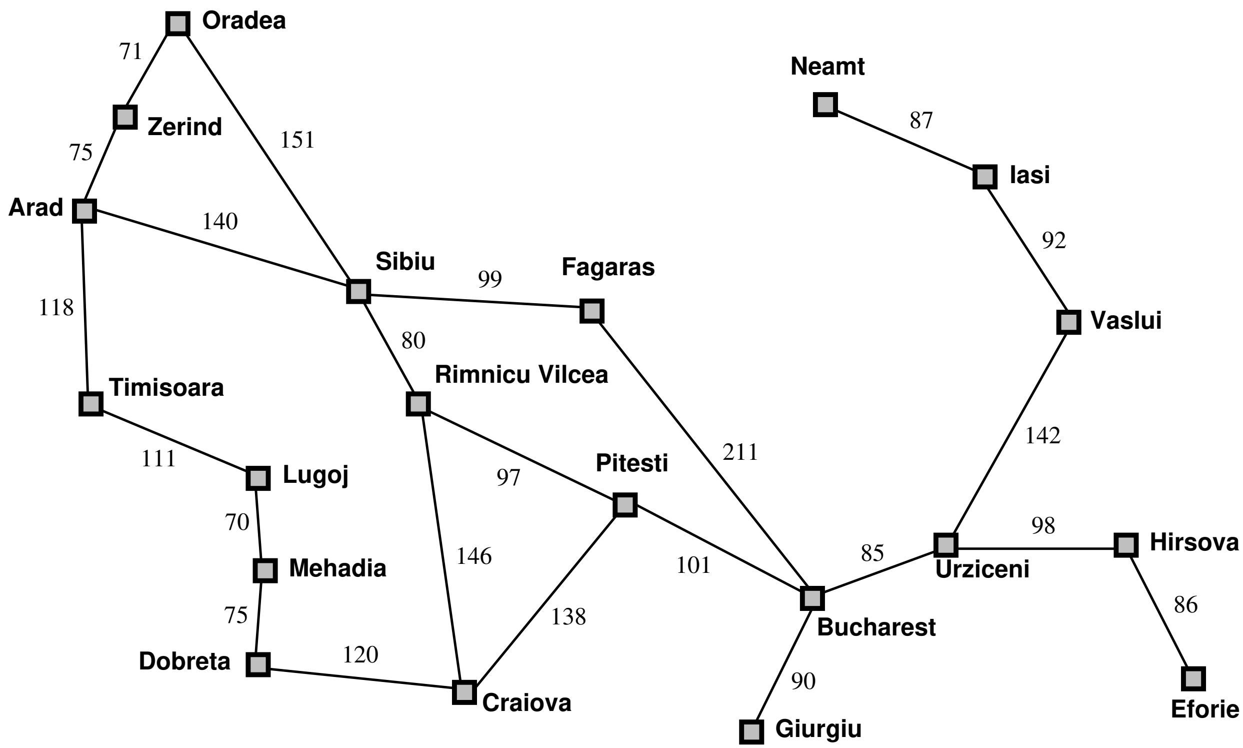
Greedy Search minimize estimated cost to reach the goal, i.e., expand the node "closest" to the goal.

A* minimize total estimated path cost to reach the goal, i.e., expand the node on the "least-cost" solution path to the goal.

Greedy Search

$h(n)$ = Estimated cost from node n to nearest goal

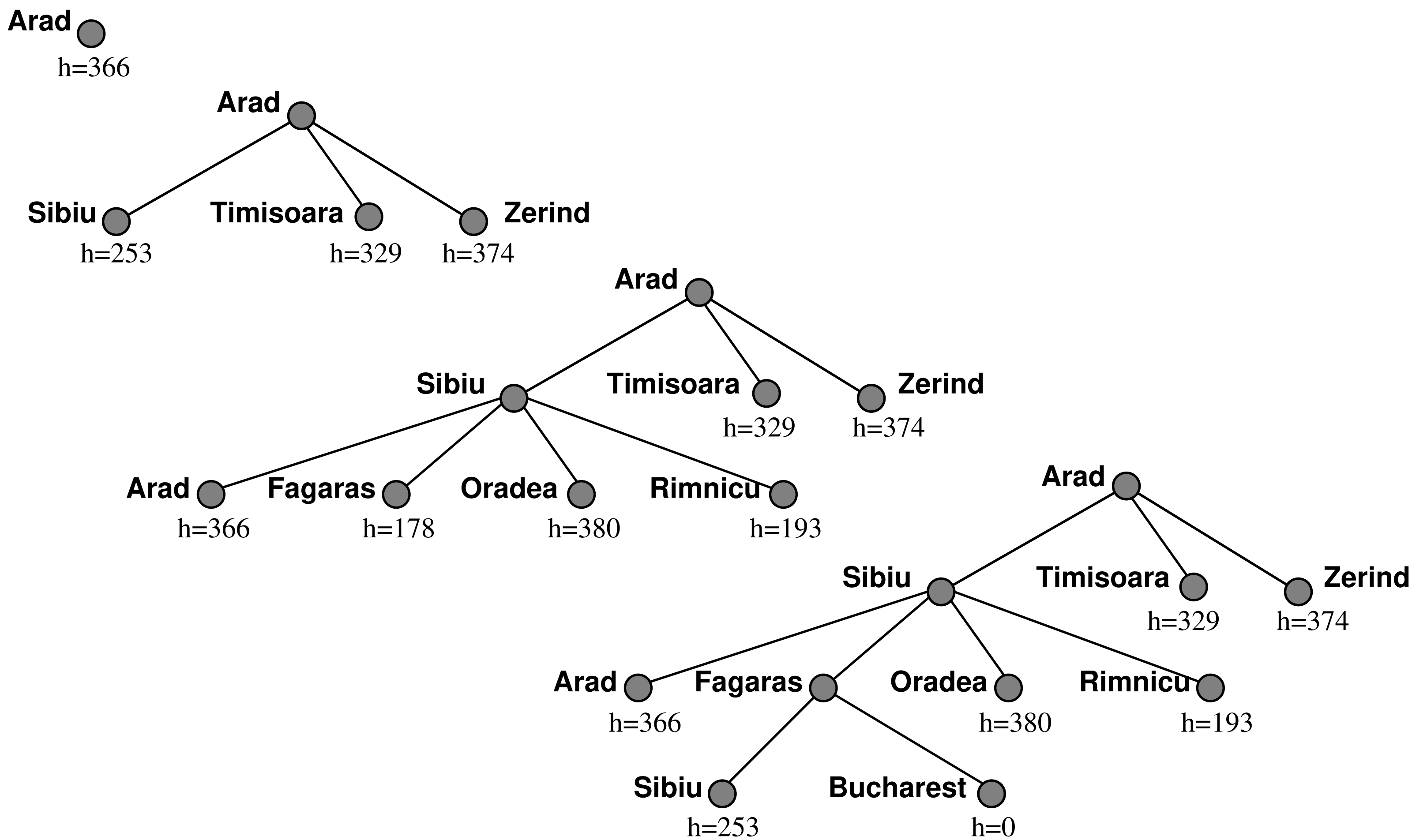
1. Set L to be the initial node(s).
2. Let n be the node on L that minimizes $h(n)$. If L is empty, fail.
3. If n is a goal node, stop and return it (and the path from the initial node to n).
4. Otherwise, remove n from L and add all of n's children to L (labeling each with its path from the initial node). Return to step 2.



Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Slide CS472–8



Arad
h=366
Arad
Sibiu
Timisoara
Zerind
h=253
h=329
h=374
Arad
Sibiu
Timisoara
Zerind
h=329
h=374
Arad
Fagaras
Oradea
Rimnicu
h=366
h=178
h=380
h=193
Arad
Sibiu
Timisoara
Zerind
h=329
h=374
Arad
Fagaras
Oradea
Rimnicu
h=366
h=380
h=193
Sibiu
Bucharest
h=253
h=0

Problem: Too Greedy

From Arad to Sibiu to Fagaras — but to Rimnicu would have been better.

Need to consider: cost of getting from start node (Arad) to intermediate nodes!

Evaluation Function for A* Algorithm

Goal:

Find the *shallowest* goal as quickly as possible.

$g(n)$ Cost of reaching node n from start node

$h(n)$ Estimated cost from node n to nearest goal

New evaluation function:

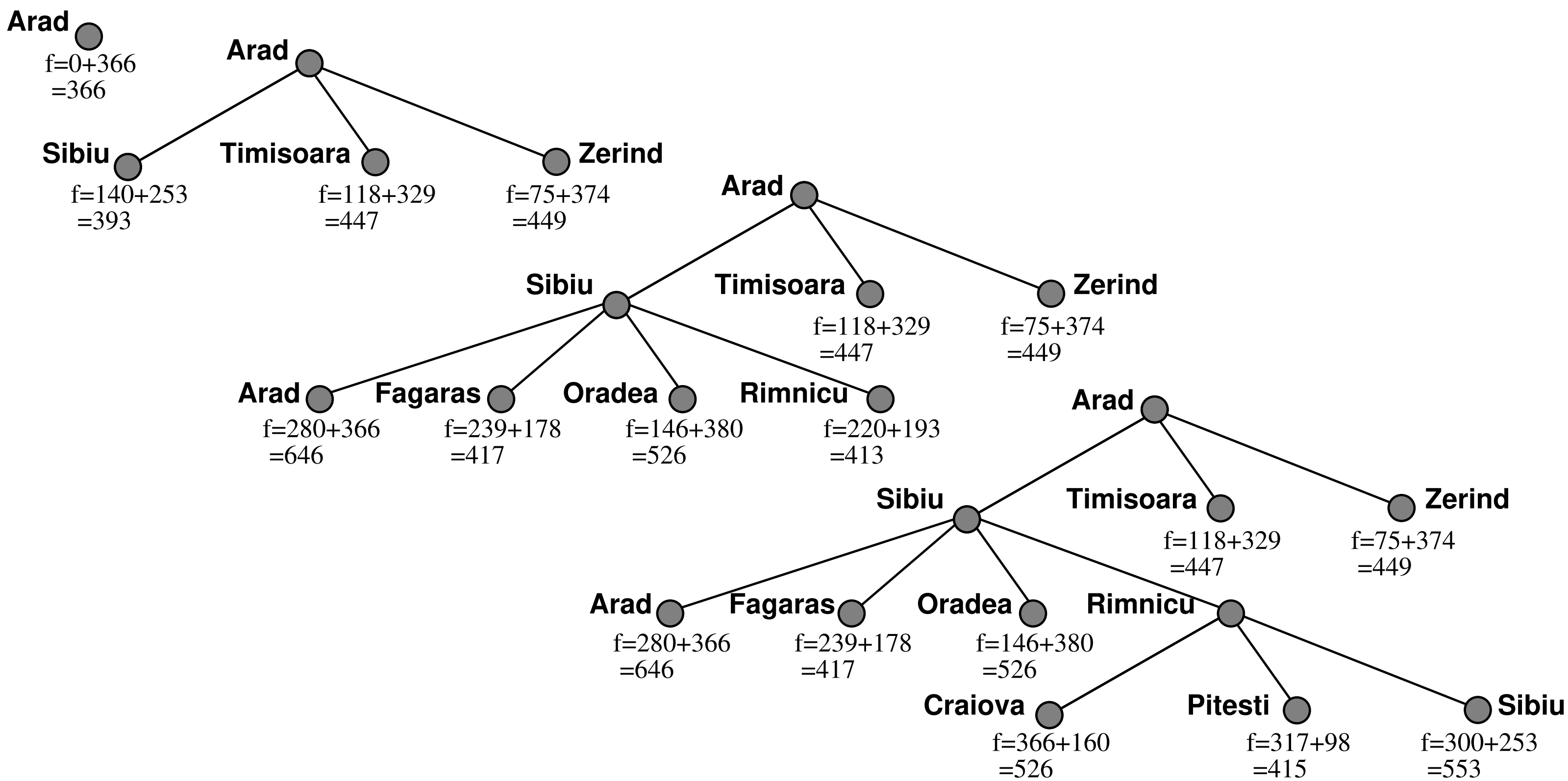
$$f(n) = g(n) + h(n)$$

$f(n)$ Estimated cost of cheapest solution through n

A*

$f(n)$ = Estimated cost of cheapest solution through n

1. Set L to be the initial node(s).
2. Let n be the node on L that minimizes to $f(n)$. If L is empty, fail.
3. If n is a goal node, stop and return it (and the path from the initial node to n).
4. Otherwise, remove n from L and add all of n's children to L (labeling each with its path from the initial node). Return to step 2.


Arad
f=0+366
=366
Arad
Sibiu
f=140+253
=393
Timisoara
f=118+329
=447
Zerind
f=75+374
=449
Arad
Sibiu
Timisoara
f=118+329
=447
Zerind
f=75+374
=449
Arad
f=280+366
=646
Fagaras
f=239+178
=417
Oradea
f=146+380
=526
Rimnicu
f=220+193
=413
Arad
Sibiu
Timisoara
f=118+329
=447
Zerind
f=75+374
=449
Arad
f=280+366
=646
Fagaras
f=239+178
=417
Oradea
f=146+380
=526
Rimnicu
Craiova
f=366+160
=526
Pitesti
f=317+98
=415
Sibiu
f=300+253
=553

Finds Optimal Path

Now expands Rimnicu ($f = (140 + 80) + 193 = 413$) over Faragas ($f = (140 + 99) + 178 = 417$).

Q. What if $h(Faragas) = 170$ (also an underestimate)?

Need Some Conditions

To guarantee that A* finds an optimal solution,

we need that h **never overestimates** the cost of reaching the goal.

Called an *admissible heuristics*.

Transfers to f, i.e., f also doesn't overestimate.

Let's also also assume (true for most admissible heuristics)
that f is **monotonic**, i.e., along any path from
the root f never decreases.

Can often modify heuristic to become monotonic.

E.g. let n be parent of n'. Suppose that
$g(n) = 3$ and $h(n) = 4$, so $f(n) = 7$.
and $g(n') = 4$ and $h(n') = 2$, so $f(n') = 6$.

But because any path through n' is also a path through n,
we can set $f(n') = 7$.

In effect: $f(n') = max(f(n), g(n') + h(n'))$.

Called the **path-max** equation.

(ignores misleading numbers in heuristic)

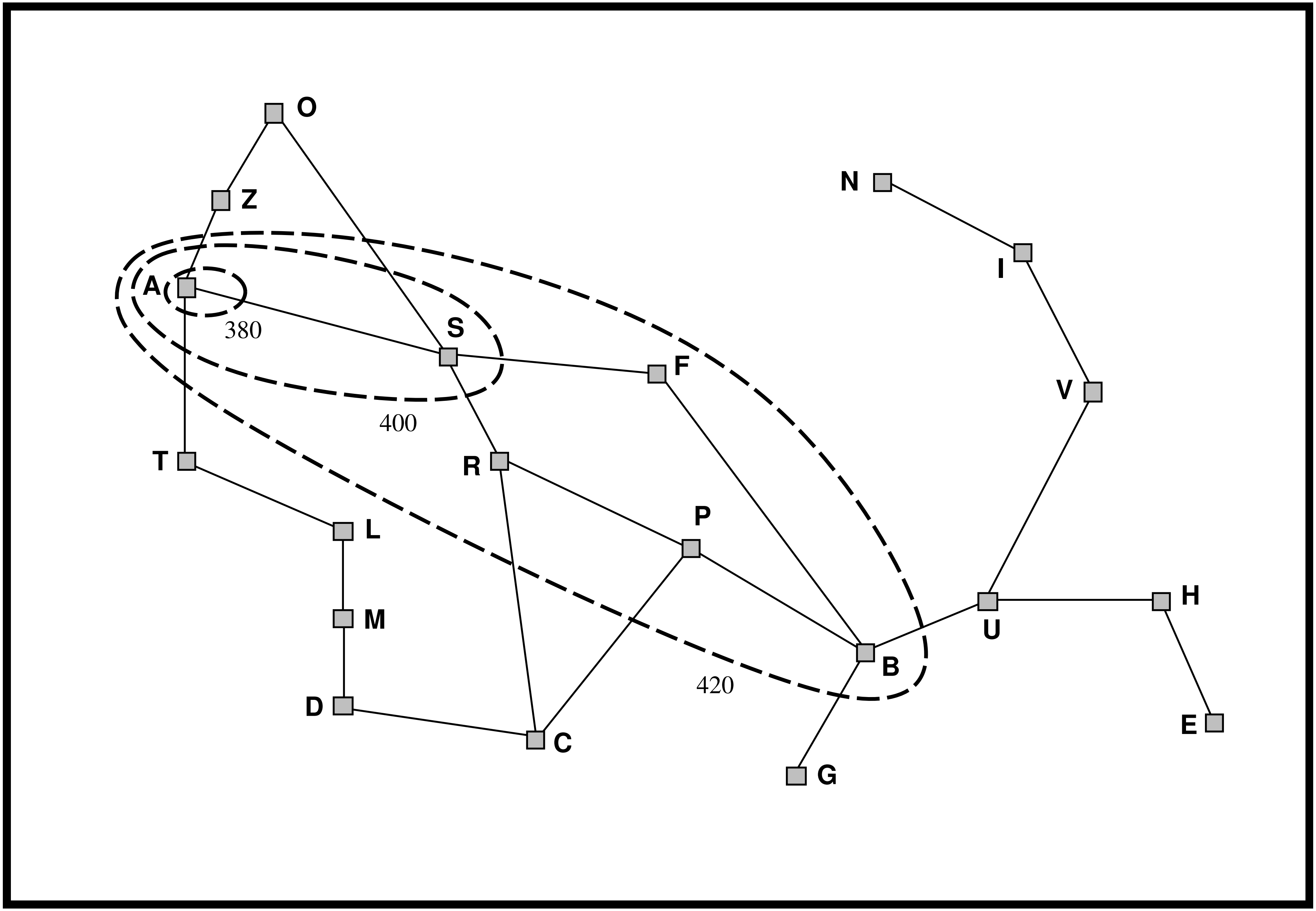
Intuition A*

Let $f^\star$ be the cost of the *optimal* solution path.

We have:

A* expands *all* nodes with $f(n) < f^\star$

A* may then expand some nodes right on "goal contour",
with $f(n) = f^\star$ before selecting a goal node.



Slide CS472–19

Observations A*

A* is **optimally efficient**: given the information in h,
no other optimal search method can expand fewer nodes.
Non-trivial and quite remarkable!

Note: A* combines information of cost to get to intermediate nodes (like "uniform cost search") with (under) estimate of cost from intermediate node to goal ("greedy search").

Aside: try uniform cost for fig. 4.5. You get "circles".
More nodes!

A*

Optimal (next)

Complete: Unless there are infinitely many nodes with $f(n) < f^\star$ Assume locally finite: (1) finite branching, (2) every operator costs at least $\delta > 0$.

Complexity: Still exponential because of breadth-first nature. Unless $|h(n) - h^\star| \leq O(log(h^\star(n))$, with $h^\star$ true cost of getting to goal.

See: IDA* (p. 106 R&N, "iterative deepening for A*".)

Proof of the optimality of A^*

f is monotonically increasing along any path from the root.

Let G be an optimal goal state, with path cost f^*
Let G_2 be a suboptimal goal state, with path cost $g(G_2) > f^*$
n is a leaf node on an optimal path to G

Because h is admissible, we must have

$$f^* \geq f(n).$$

Also, if n is not chosen over G_2, we must have

$$f(n) \geq f(G_2).$$

Gives us $f^* \geq f(G_2) = g(G_2)$. (Then G_2 is *not* suboptimal!)

Heuristic Functions: Example

5	4	
6	1	8
7	3	2

Start State

1	2	3
8		4
7	6	5

Goal State

8-puzzle

1. h_C = number of misplaced tiles
2. h_M = Manhattan distance

Admissible?

Which one should we use?

Importance of $h(n)$

$$h_C \leq h_M \leq h_{\text{opt}}$$

Prefer h_M.

Note: Expand all nodes with $f(n) = g(n) + h(n) < f^\star$

So, $g(n) < f^\star - h(n)$, higher h means fewer n's.

Aside: How would we get an h_{opt} ?

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Inventing Heuristics

Automatically

A tile can move from sq A to sq B if

A is adjacent to B and B is blank.

(a) A tile can move from sq A to sq B if A is adjacent to B.

(b) A tile can move from sq A to sq B if B is blank.

(c) A tile can move from sq A to sq B.

If all admissible, combine them by taking the *max*.