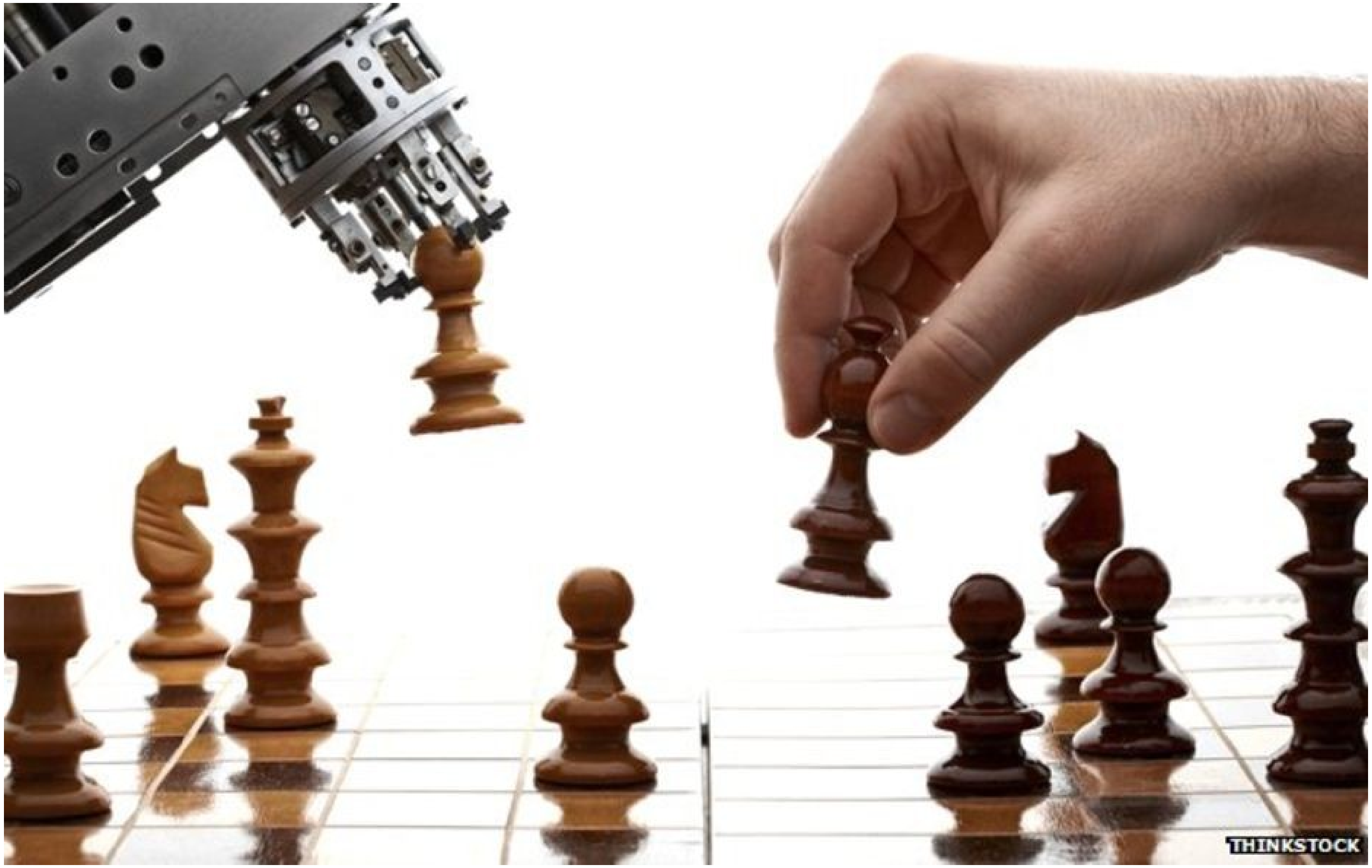


Introduction to Artificial Intelligence

COSC 4550 / COSC 5550

Professor Cheney
9/13/17

heuristic (informed) search!

rather than treating graphs/trees as abstract structures,
let's use the fact that we know something about
the state at each node

expand the node that seems most promising,
based on some “heuristic” function
(approximation for the quality of that state)

looking only one step ahead leads to a
“greedy” algorithm

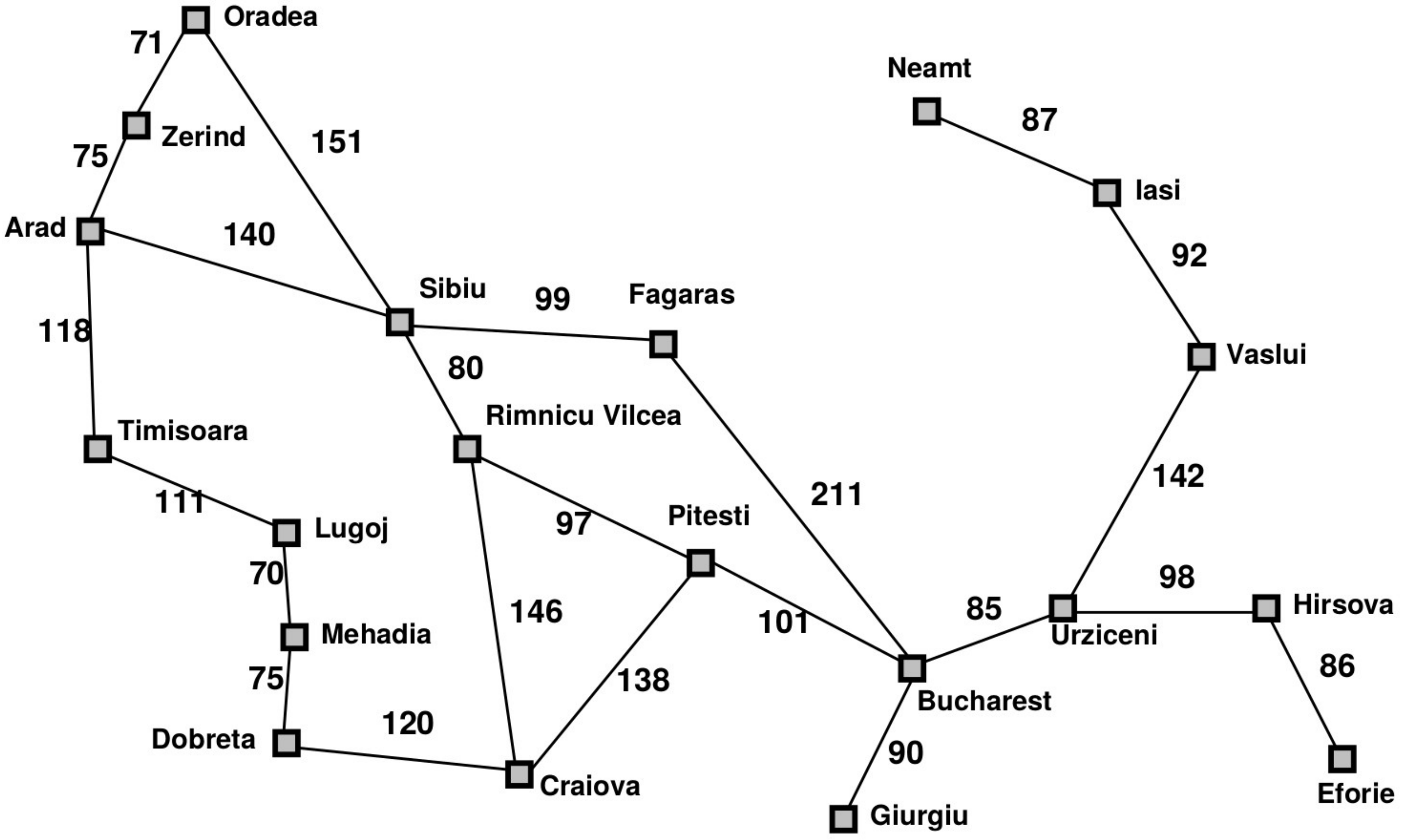
0) frontier: Arad (h=**366**)

1) frontier: A→Zerind (h=374), A→Sibiu (h=**253**), A→Timisoara (h=329)

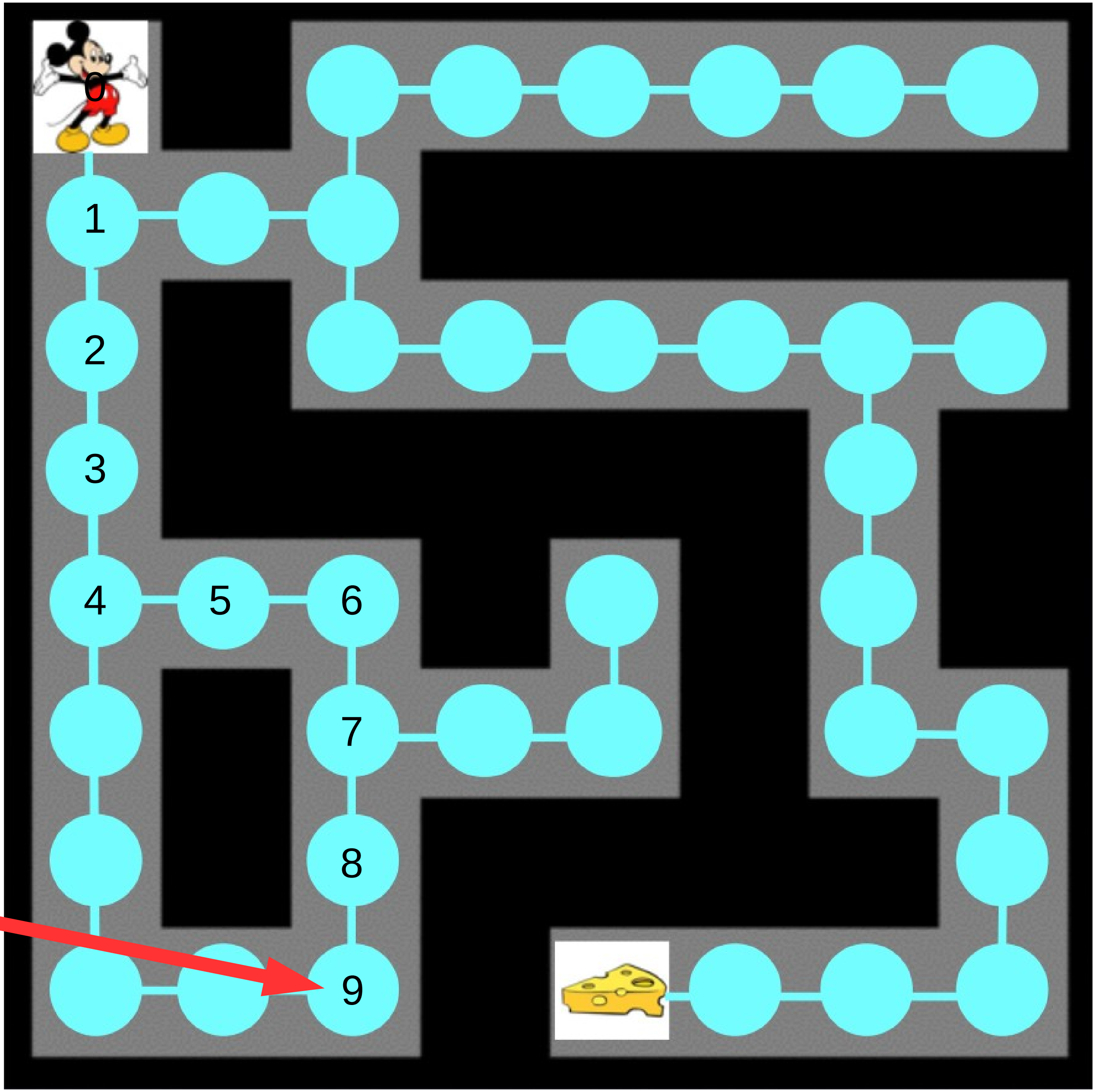
2) frontier: A→Z (h=374), A→T (h=329), A→S→F (h=**178)**, A→S→RV (h=193)

3) frontier: A→Z (h=374), A→T (h=329), A→S→RV (h=193), A→S→F→**B (h=0)**

heuristic function, h(x):



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



High value for heuristic function,

but not a promising node to get to the goal!

having a good heuristic function
is of the utmost importance!

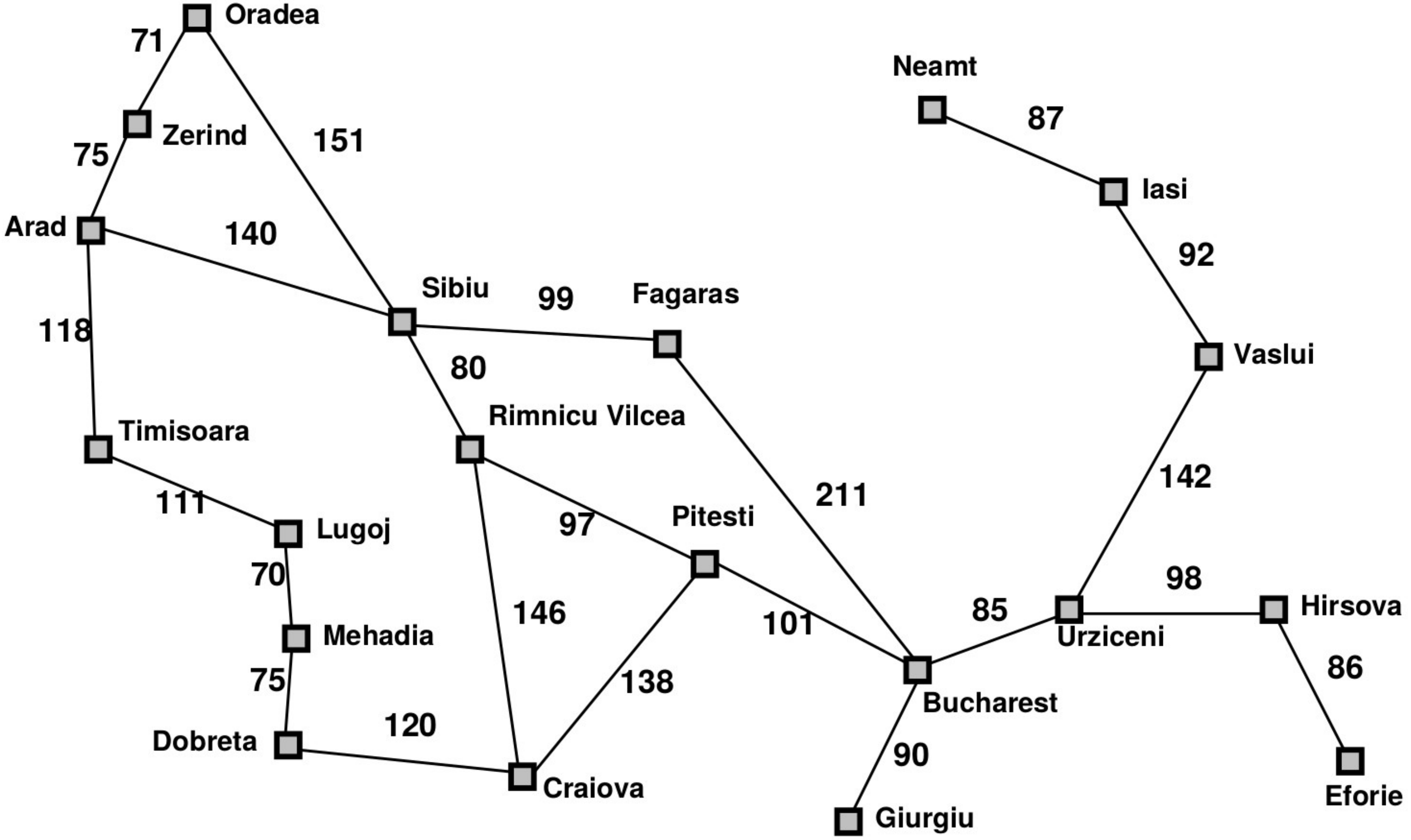
having a bad one can be even worse than uninformed search
(leading to deceptive local optima)

many modern machine learning agents try to learn
this function from experience

wait a sec… the solution we got from heuristic search was not the optimal solution we found earlier (from uniform cost search)

A→ S→ F→B = 140 + 99 + 211 = 450
A→S→RV→P→B = 140 + 80 + 97 + 101 = 418

what went wrong???



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

A* search

select nodes for expansion based on the estimated total cost

this is from both your experienced costs so far: $g(x)$
and your (heuristic for) estimated future costs: $h(x)$

0) frontier: Arad (g=0, h=366, c=**366**)

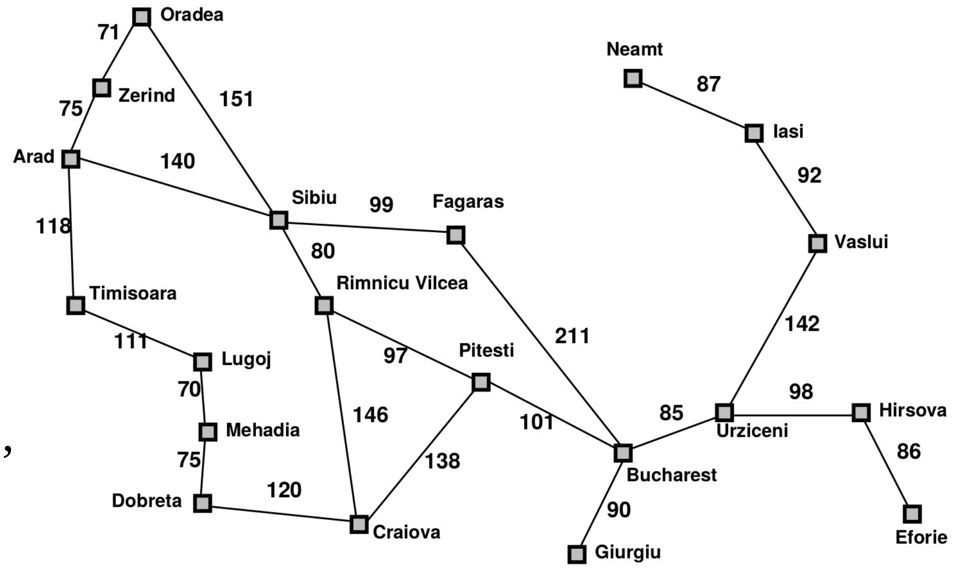
1) frontier: A→Z (g=140,h=374,c=514),
A→S (g=140,h=253,c=**393**),
A→T (g=140,h=329,c=469)

2) frontier: A→Z (140+374=514),
A→T (140+253=393),
A→S→F (h=140+99+178=417),
A→S→RV (140+80+193=**413**)

3) frontier: A→Z (140+374=514),
A→T (140+253=393),
A→S→F (h=140+99+178=417),
A→S→RV→P (140+80+97+98=**415**)

4) frontier: A→Z (140+374=514),
A→T (140+253=393),
A→S→F (h=140+99+178=**417**),
A→S→RV→P→B (140+80+97+101=418)

5) frontier: A→Z (140+374=514),
A→T (140+253=393),
A→S→F→B (h=140+99+211=450),
A→S→RV→P→B (140+80+97+101=**418**)



Straight−line distance to Bucharest

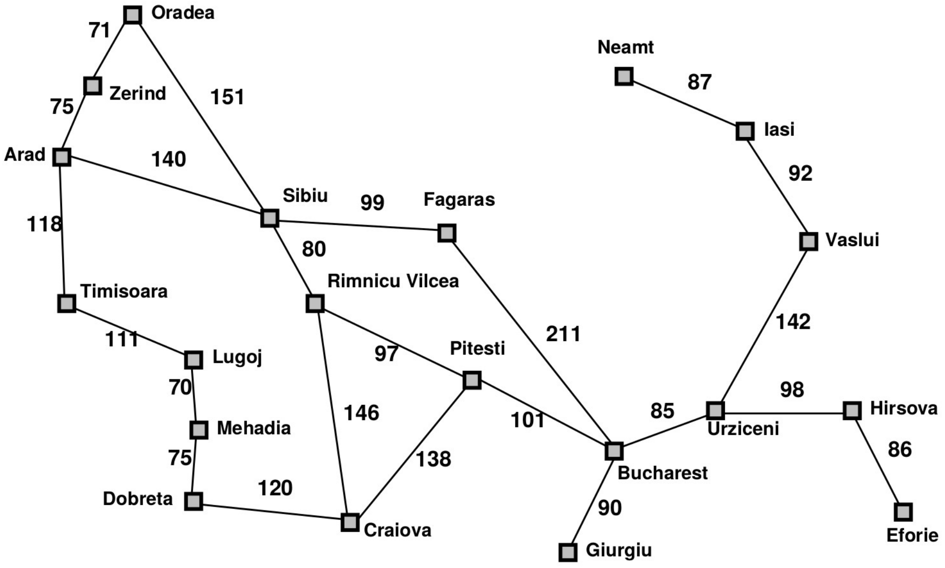
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

stop when you try
to expand goal node!

(expanding a path
means it's the lowest
cost path to that node)

how could this go wrong?

what if our heuristic estimate of the cost from Arad to Sibiu was 300?

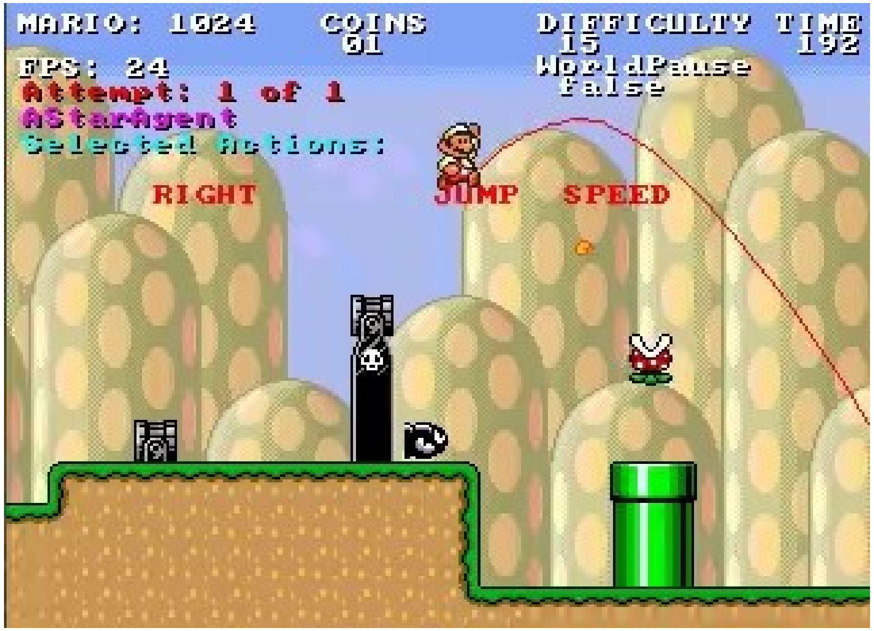


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

we would go A→Z→O→S (since $75 + 71 + 151 = 297 < 300$)

our heuristic function h(x) must be optimistic!!!
(that way we always increase costs by expanding paths)
(we have an incentive to explore new paths we're unsure about)

if so, A* is optimal (and widely used)!


MARIO: 1024
COINS
01
DIFFICULTY
15
TIME
192
FPS: 24
WorldPause
false
Attempt: 1 of 1
AStarAgent
Selected Actions:
RIGHT
JUMP
SPEED

A* Mario

exploration vs. exploitation

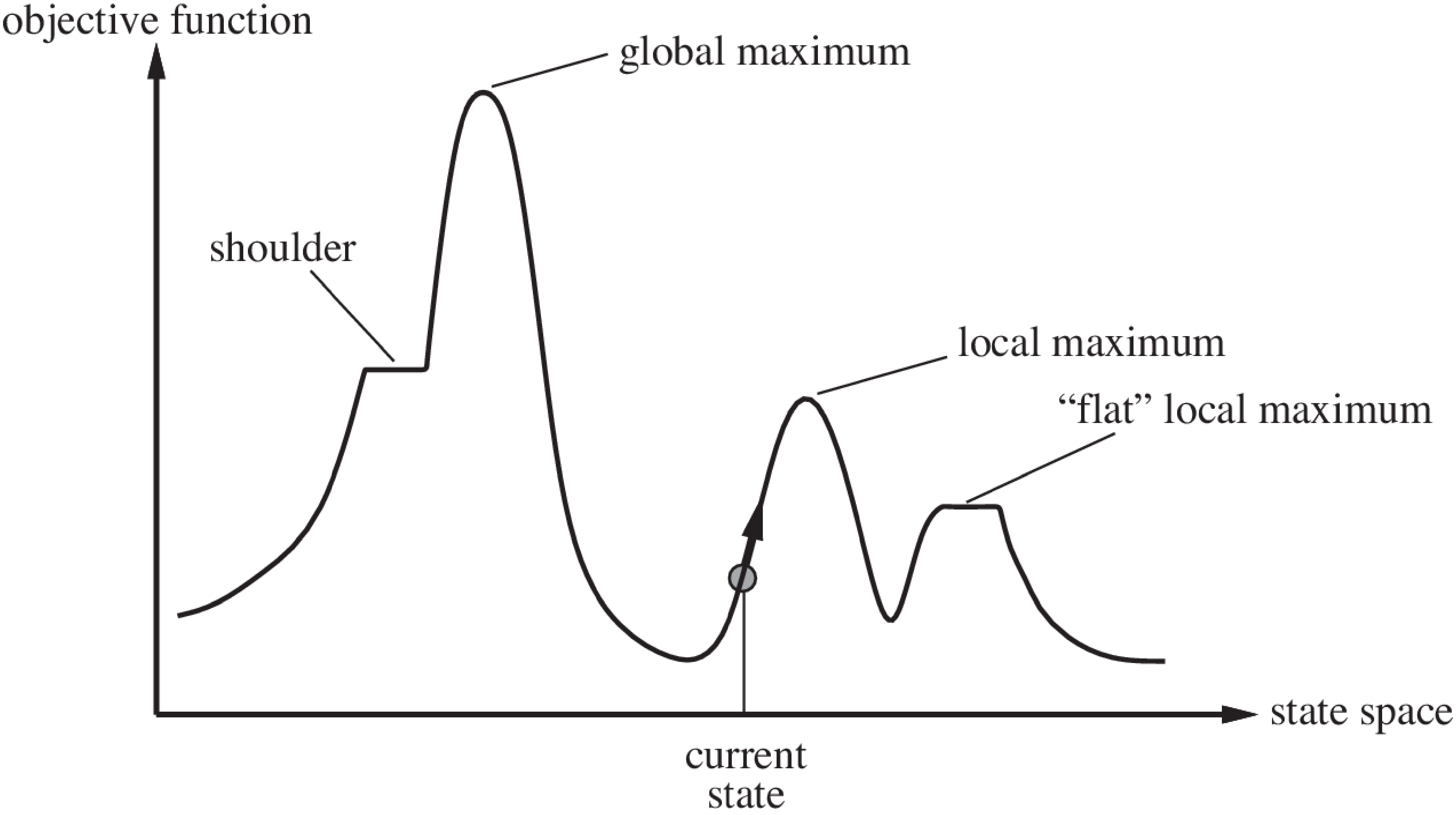
should we follow the path (or use the solution)
that we know works pretty well?

or should we try a path that we haven't use
before, in hopes that it could be even better?

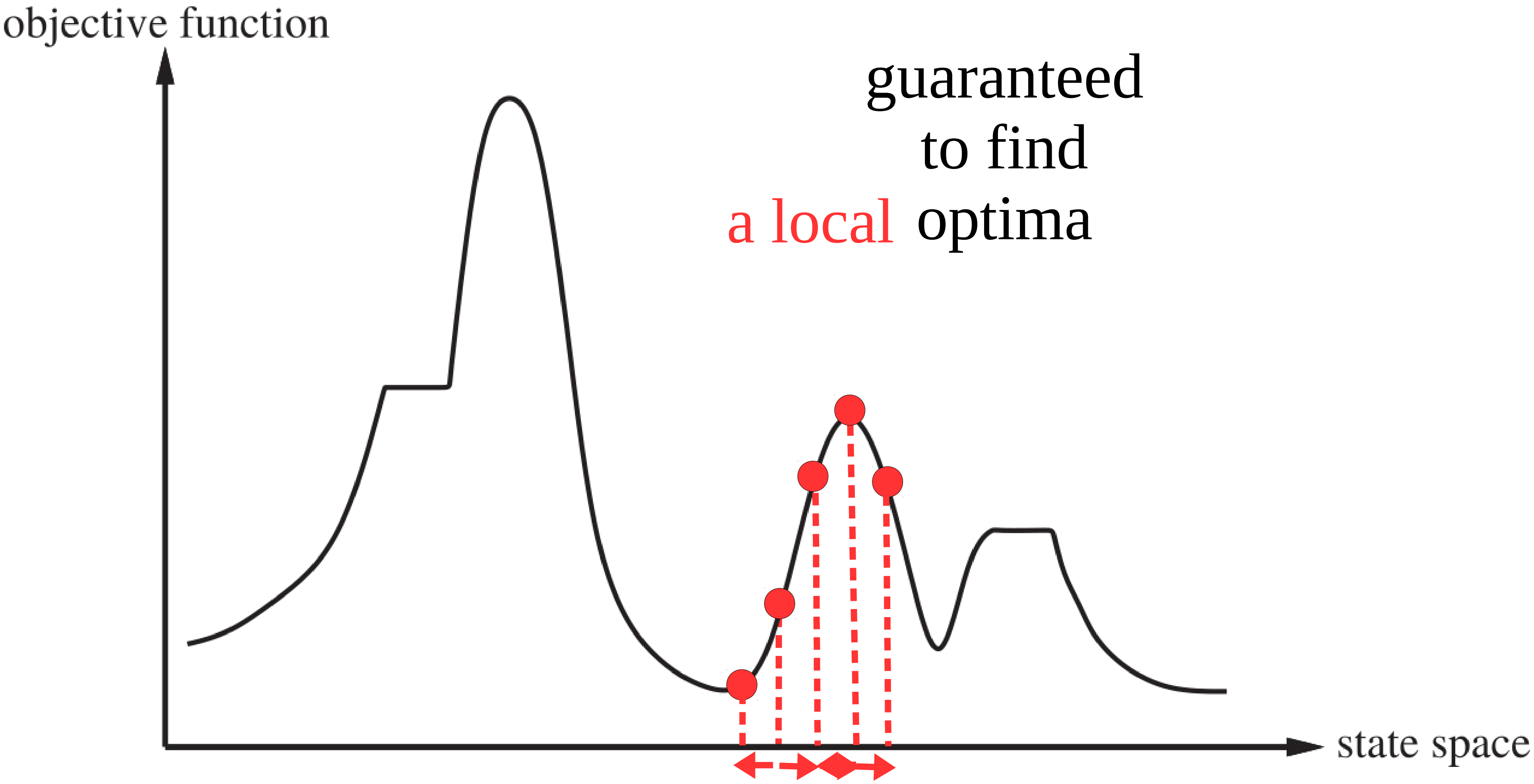
optimization!

With this model, we apply the same simple rule as before:

fitness (state space) landscape


objective function
global maximum
shoulder
local maximum
"flat" local maximum
state space
current
state

hillclimbing



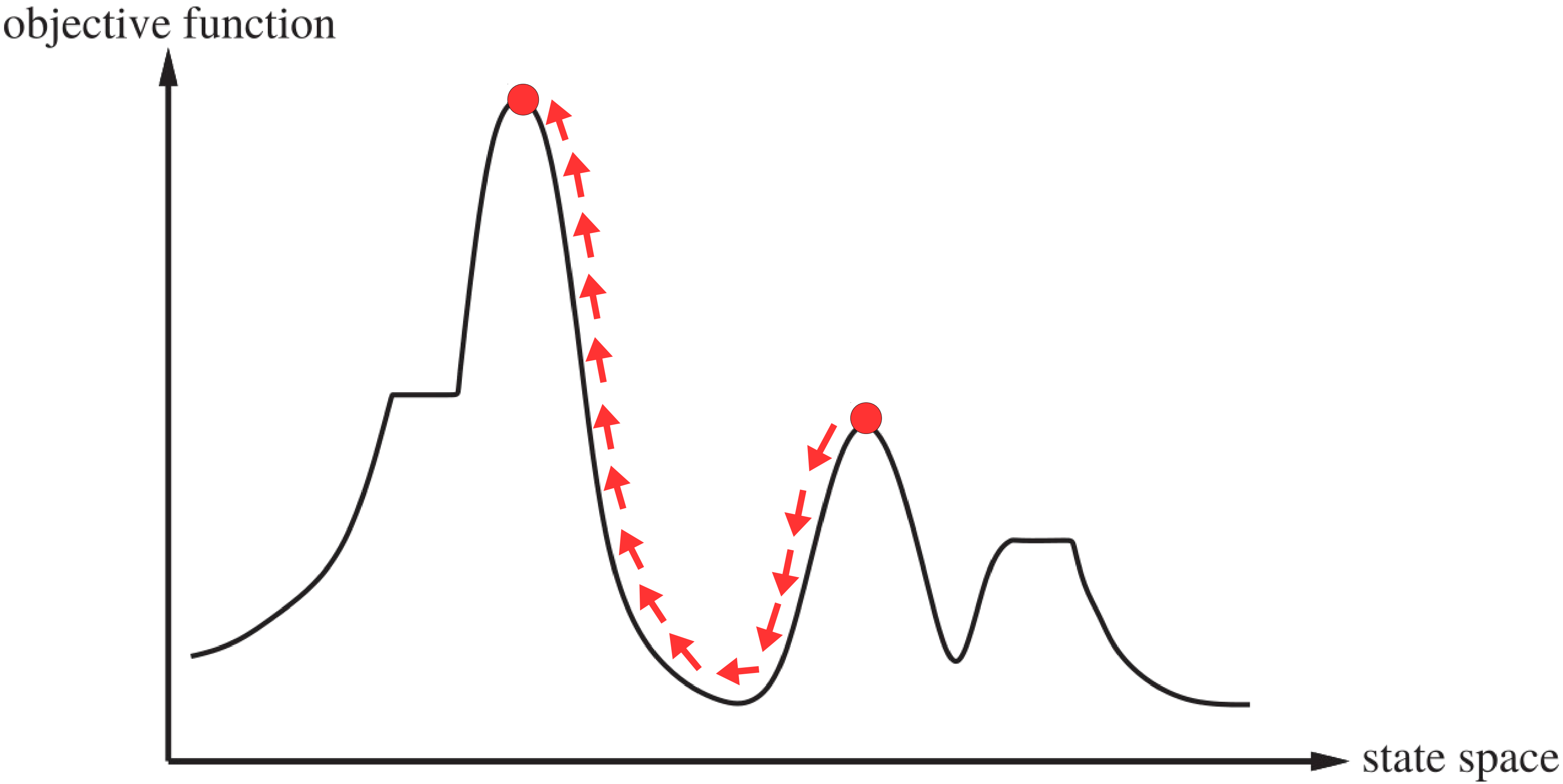
guaranteed
to find
a local optima

but wouldn't we rather have a global optima?

to do so, we'd have to accept (many) negative mutations to get to the better "fitness peak" hillclimbing is greedy local search

very simply to code and understand:

if your new state is better, keep it
if your new state is worse,
throw it away and keep your old state


objective function
state space

what if just sometimes,
we accepted negative mutations?