

Heuristic Search

Scott Wallace

CS 440

WSU Vancouver

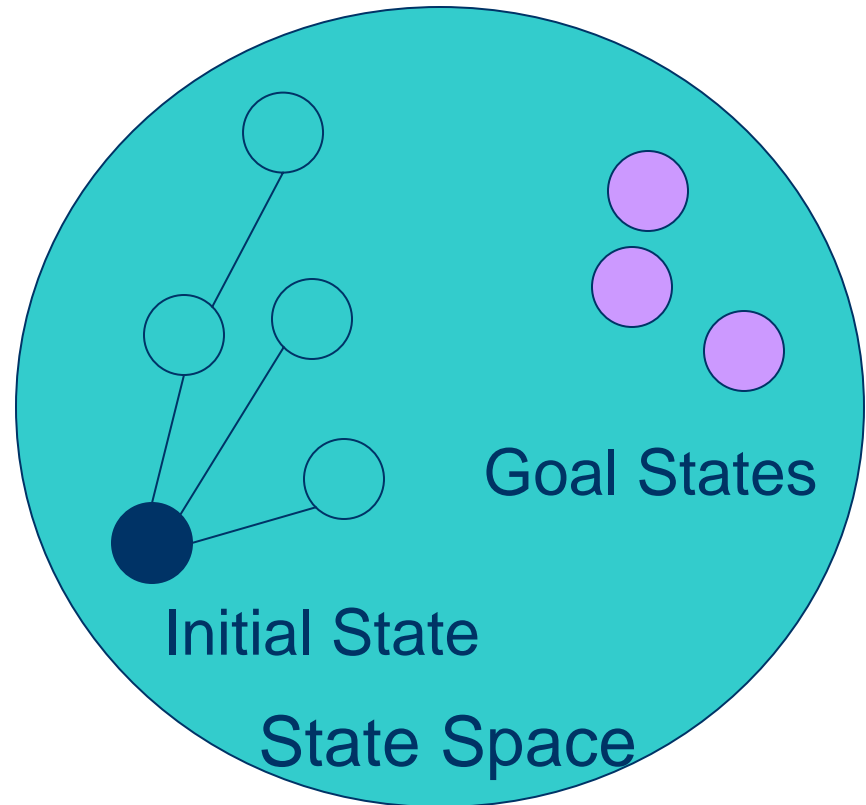


Outline

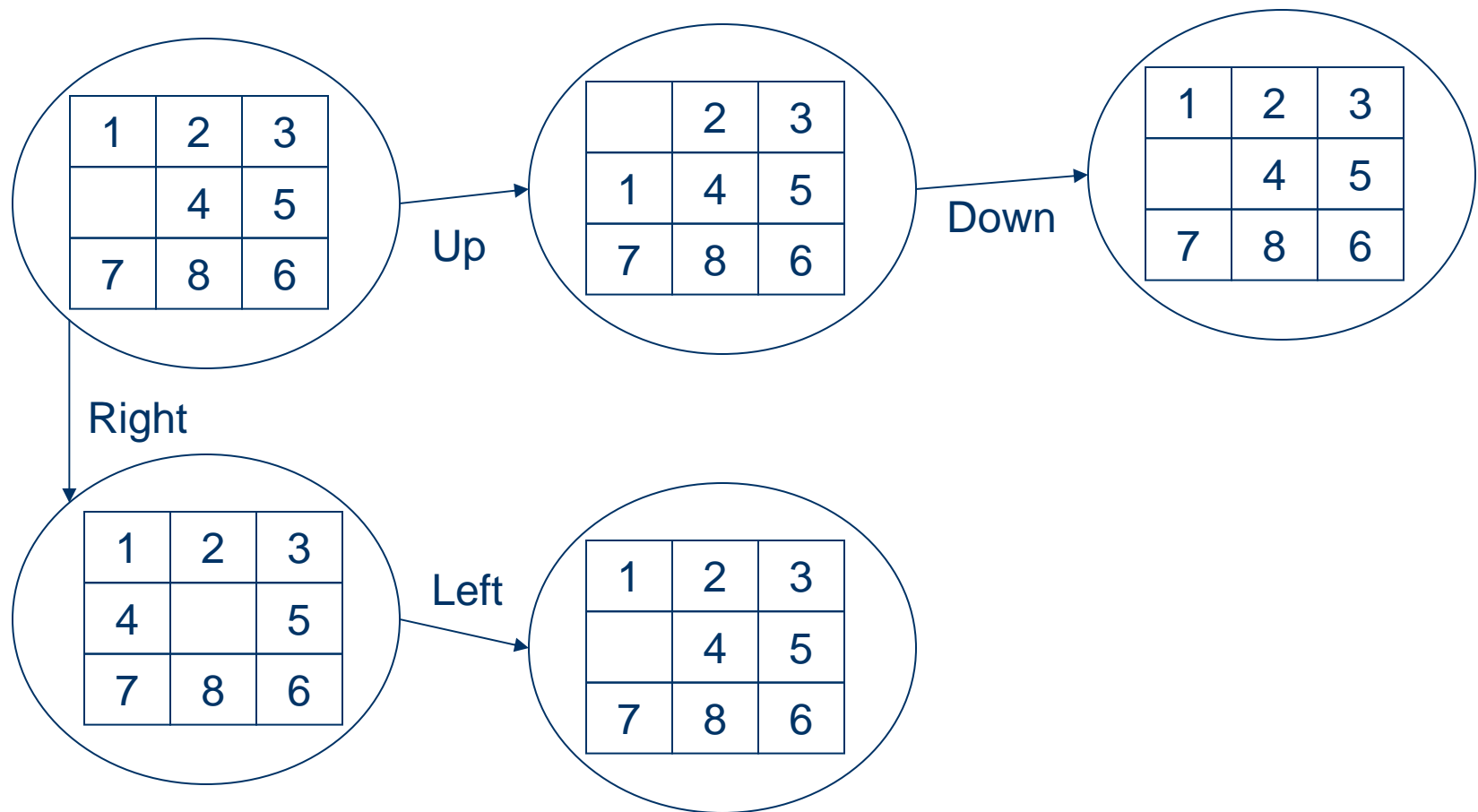
- Heuristic Functions
- Greedy Search (Greedy Best First)
- A* Search
- A* Variants
- Creating Heuristics (Chapter 4.2)

State Space Search

- Potential world states are contained in a set
- Path (solution) leads through the state space



Eight Puzzle Search Tree



Tree Search

TreeSearch(*problem*, *strategy*)

initialize search tree with initial state of *problem*

loop

if no candidates for expansion, **return** *failure*

choose a leaf for expansion based on *strategy*

if node corresponds to a goal state,

return *solution*

else expand node and add successors to tree

Recall Uniform Cost Search

- Open list is a priority queue
 - Nodes ordered from lowest cost (high priority) to highest cost (low priority)
 - Ensures we expand in intervals of equal cost
 - Usually better than BFS
 - Optimal and Complete

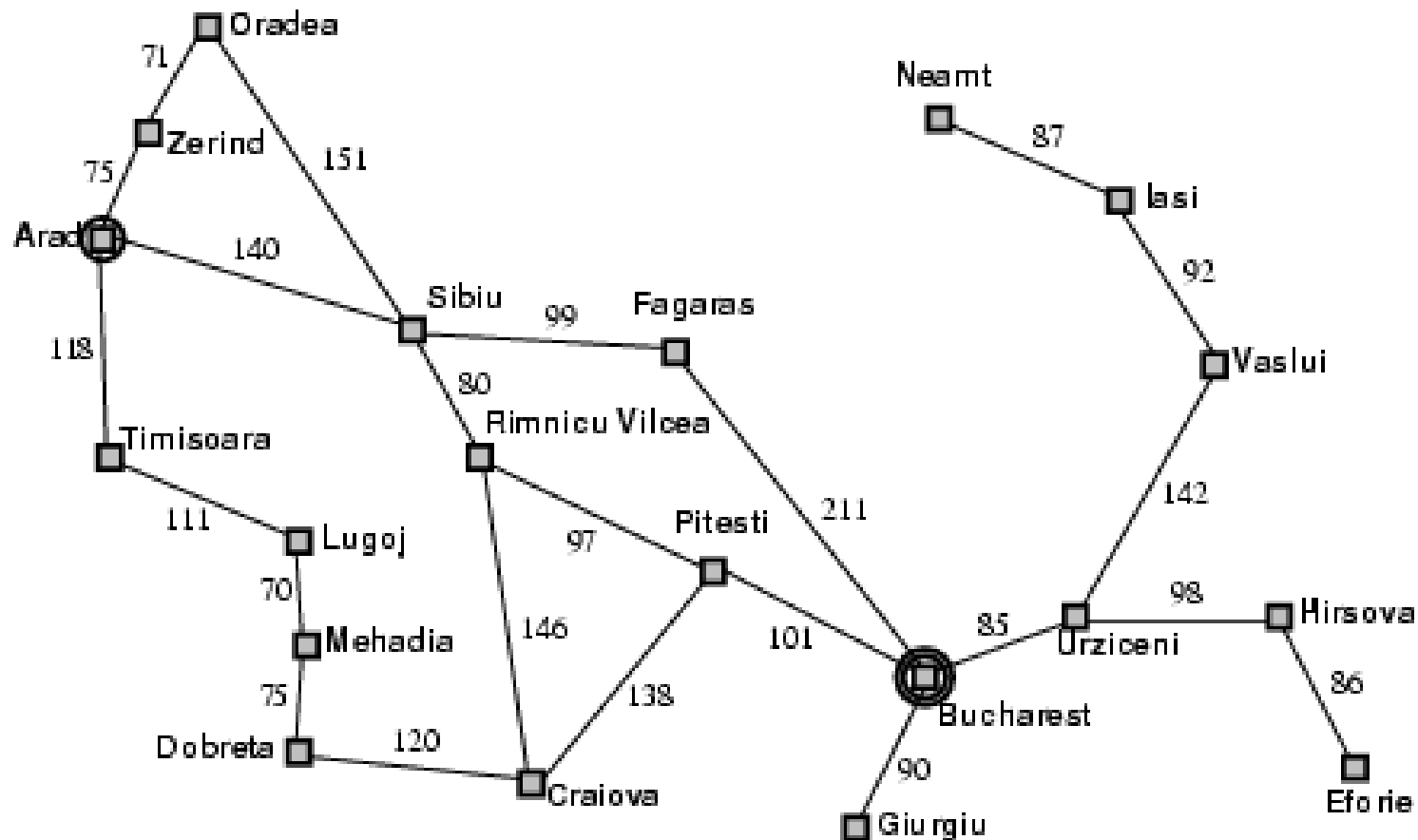
Heuristic Functions

- In Uniform Cost Search, we apply a function to a node in the search tree to determine its cost (the path cost).
- **Heuristic functions** work the same, but instead of accumulating cost thus far, they **estimate** the remaining cost to be incurred.

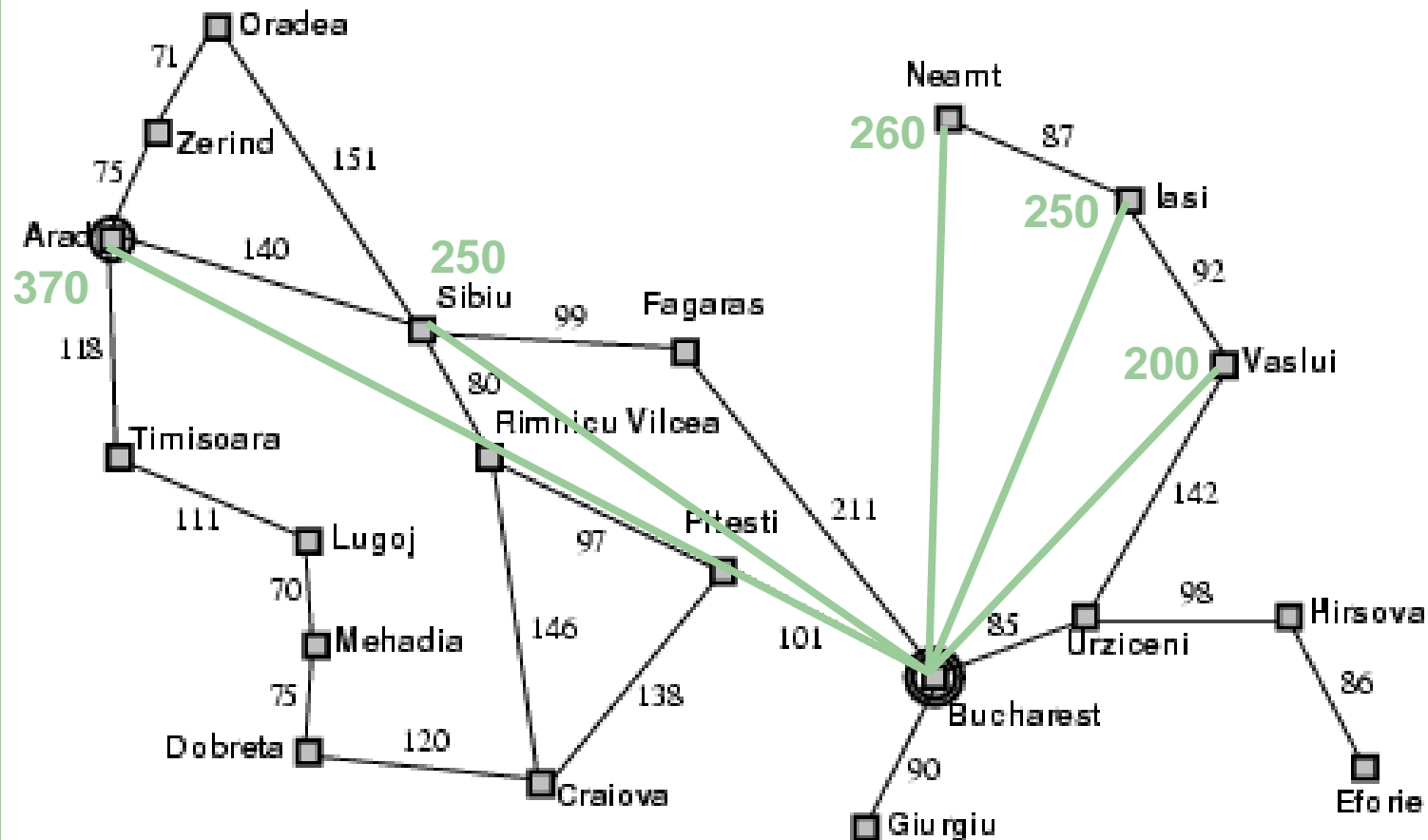
Heuristic Functions II

- Can't make a 'general' heuristic
- Heuristics require knowledge about the domain (task)
- Heuristics are at the heart of **informed search** methods

Route Finding



Route Finding (Heuristics)



Functions thus far

$g(n)$ cost of path so far (as in UCS)

$h(n)$ estimated remaining cost (heuristic)

$d(n)$ the depth of a node in the search tree

$f(n)$ the function used to order the open list
(a min queue, so low values are better)

Uninformed Search with P. Queue

- Breadth First: $f(n) = d(n)$
- Depth First: $f(n) = 1 / d(n)$
- Uniform Cost: $f(n) = g(n)$

Informed search will use $h(n)$

Greedy Search (Greedy Best First)

- $f(n) = h(n)$
- The path that looks like its closest to the goal is tested first
- Somewhat like DFS, but with a directionality component

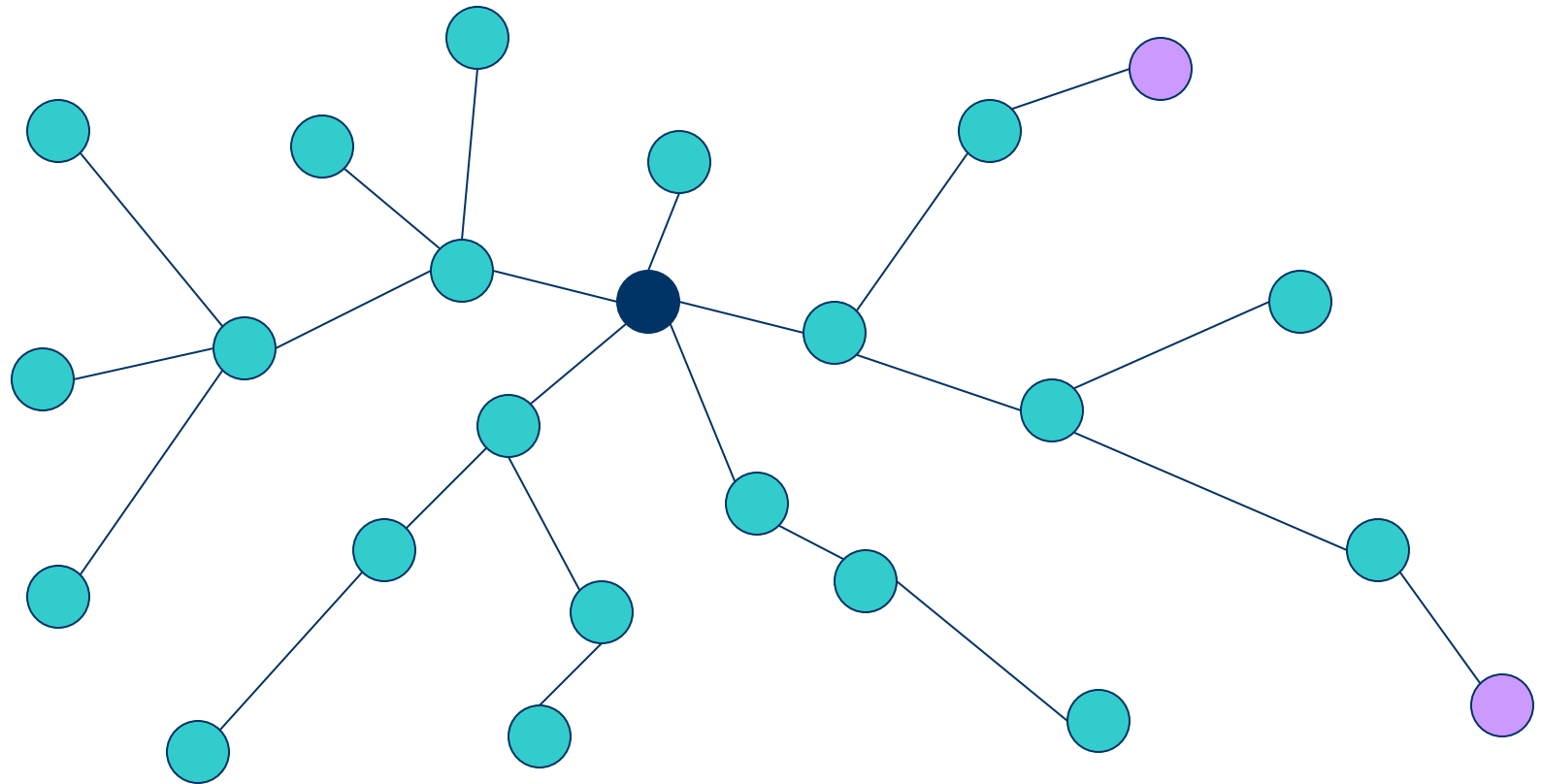
Greedy Search

- Optimal: No
- Complete: No
- Time and Space Complexity:
 - Depend on interaction of heuristic and domain

A* Search

- $f(n) = g(n) + h(n)$
- High priority path is the one that looks like it will end up being the lowest cost in the end
- Incorporates aspects of Greedy and UCS
 - behaves like UCS, but is biased towards paths that look like they're close to the goal

Order of Expansion



A*

- Optimal: Yes
- Complete: Yes
- Optimality and Completeness rely on heuristic
 - Must be admissible

Inadmissible Heuristics

- If heuristic overestimates:
 - Best path may look really bad
 - Will be put on back of queue
 - A better looking (but actually sub-optimal) solution may be found first

A* and Graph Search

- May find duplicate paths to same node.
- Need to be careful here:
 - 2nd path may actually be cheaper than 1st path
 - Implications for closed list
 - Monotonic (consistent) heuristics avoid this issue
 - $h(n) \leq c(n,a,n') + h(n')$
 - If $h(n)$ is consistent, $f(n)$ is monotonically increasing

A* Variants

- A* is quite useful
- Memory can still be an issue
 - IDA* (iterative deepening A*)
 - SMA* (simplified memory bounded A*)
- Memory constraints can create issues similar to thrashing