Basics of AI and Machine Learning

State-Space Search: Uniform Cost Search

Jendrik Seipp

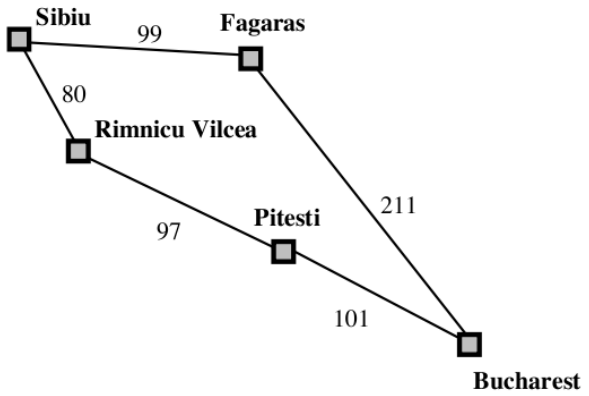
Linköping University

Slides modified from Basel AI group, with permission

Introduction

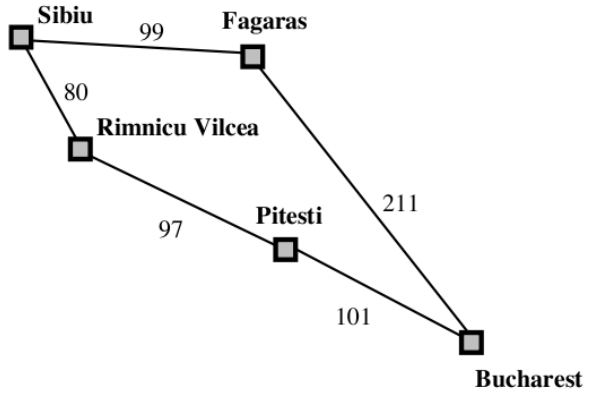
Uniform Cost Search

- breadth-first search optimal if all action costs equal
- otherwise no optimality guarantee $\rightsquigarrow$ example:



Uniform Cost Search

- breadth-first search optimal if all action costs equal
- otherwise no optimality guarantee $\rightsquigarrow$ example:



remedy: uniform cost search

- always expand a node with minimal path cost (n.path_cost a.k.a. $g(n)$)
- implementation: priority queue (min-heap) for open list

Algorithm

Uniform Cost Search

Uniform Cost Search

```
open := new MinHeap ordered by g
open.insert(make_root_node())
closed := new HashSet
while not open.is_empty():
    n := open.pop_min()
    if n.state ∉ closed:
        closed.insert(n.state)
        if is_goal(n.state):
            return extract_path(n)
        for each ⟨a, s'⟩ ∈ succ(n.state):
            n' := make_node(n, a, s')
            open.insert(n')
return unsolvable
```

Uniform Cost Search: Discussion

- as in BFS-Graph, a set is sufficient for the closed list
- a tree search variant is possible, but rare
- identical to Dijkstra's algorithm for shortest paths

Properties

Completeness and Optimality

properties of uniform cost search:

- uniform cost search is complete:
  will eventually exhaust whole search space
- uniform cost search is optimal:
  expands nodes by increasing path cost

Summary

Summary

uniform cost search: expand nodes in order of ascending path costs

- usually as a graph search
- then corresponds to Dijkstra's algorithm
- complete and optimal