Artificial Intelligence Foundation – JC3001

Lecture 7: Search II: Informed Search II

Prof. Aladdin Ayesh (aladdin.ayesh@abdn.ac.uk)

Dr. Binod Bhattarai (binod.bhattarai@abdn.ac.uk)

Dr. Gideon Ogunniye, (g.ogunniye@abdn.ac.uk)

September 2025

Material adapted from:
Russell and Norvig (AIMA Book): Chapter 3 (3.5–3.6)
Malte Helmert (University of Basel)

Course Progression

- Part 1: Introduction
  1. Introduction to AI ✓
  2. Agents ✓
- Part 2: Problem-solving
  1. Search 1: Uninformed Search ✓
  2. **Search 2: Heuristic Search**
  3. Search 3: Local Search
  4. Search 4: Adversarial Search
- Part 3: Reasoning and Uncertainty
  1. Reasoning 1: Constraint Satisfaction
  2. Reasoning 2: Logic and Inference
  3. Probabilistic Reasoning 1: BNs
  4. Probabilistic Reasoning 2: HMMs
- Part 4: Planning
  1. Planning 1: Intro and Formalism
  2. Planning 2: Algos and Heuristics
  3. Planning 3: Hierarchical Planning
  4. Planning 4: Stochastic Planning
- Part 5: Learning
  1. Learning 1: Intro to ML
  2. Learning 2: Regression
  3. Learning 3: Neural Networks
  4. Learning 4: Reinforcement Learning
- Part 6: Conclusion
  1. Ethical Issues in AI
  2. Conclusions and Discussion

Objectives

- Informed Search
  - Greedy Best First Search ✓
  - A* Search
- Heuristic Functions

Outline

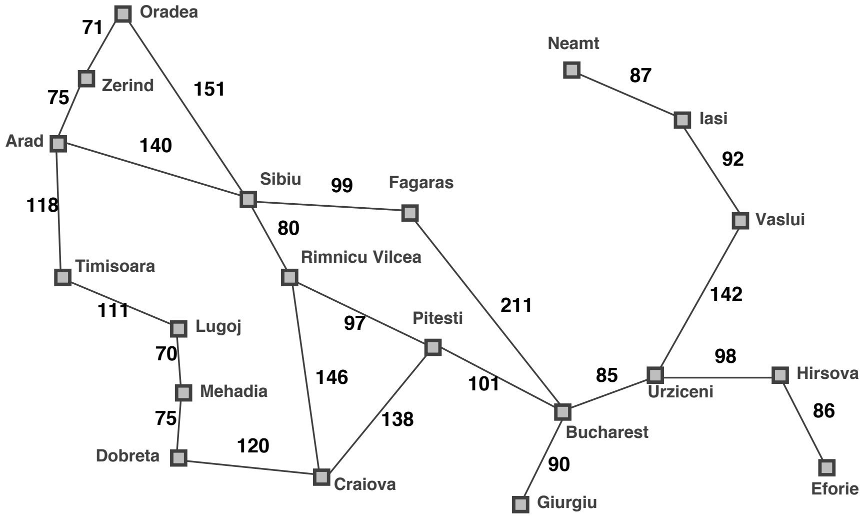
- Informed Search
- Heuristic Functions

A* Search

- The A* search is probably the most used type of heuristic search
- It combines the cost to reach a node ($g(n)$) with the cost to get from the node to the goal ($h(n)$)
  $f(n) = g(n) + h(n)$
- $f(n)$ is the **estimated cost** of the cheapest solution through n
- Since we are trying to find the cheapest solution, it makes sense to try the node with the lowest $f(n)$ first
- Under certain conditions of $h(n)$, A* search is complete and optimal

```
function aStarSearch(problem $p$, heuristic $h$)
    $closed \leftarrow \{\}$                                ▷ We keep a collection of explored states
    $frontier$.add$(newNode(p.initial), 0)$
    loop
        if $frontier$ is empty then
            return $fail$
        $n \leftarrow frontier$.get
        $closed$.add$(n.state)$                             ▷ Where we keep states we already visited
        if p.goalTest$(n.state)$ then
            return getPath$(n)$
        for all $a \in p$.actions$(n)$ do
            $n' \leftarrow a$.result$(n.state)$
            if $n'.state \notin closed$ then
                $v \leftarrow n'.cost + h(n'.state)$        ▷ Order nodes by full path cost estimate
                $frontier$.add$(n', v)$                     ▷ And only explore states we haven't visited
```

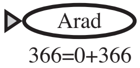
A* Search Example



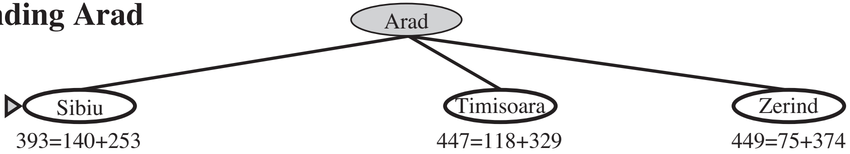
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* Search Example

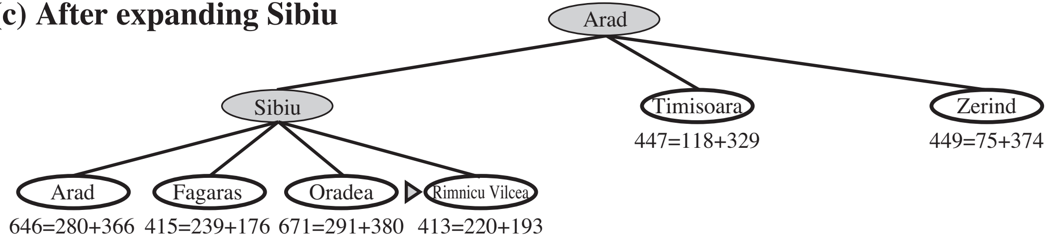
(a) The initial state



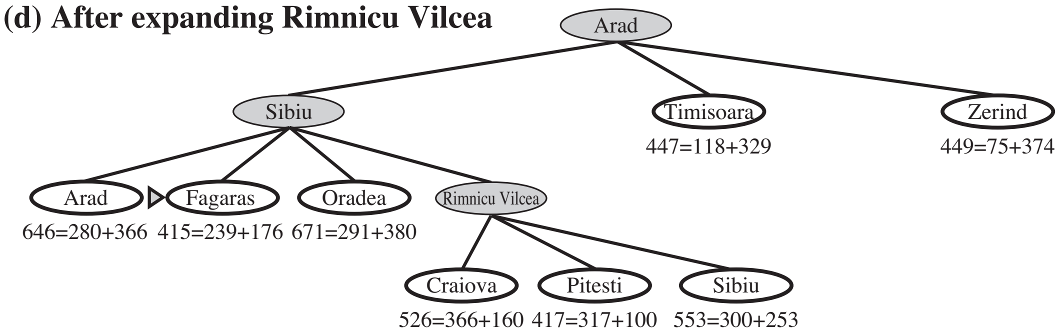
(b) After expanding Arad



(c) After expanding Sibiu

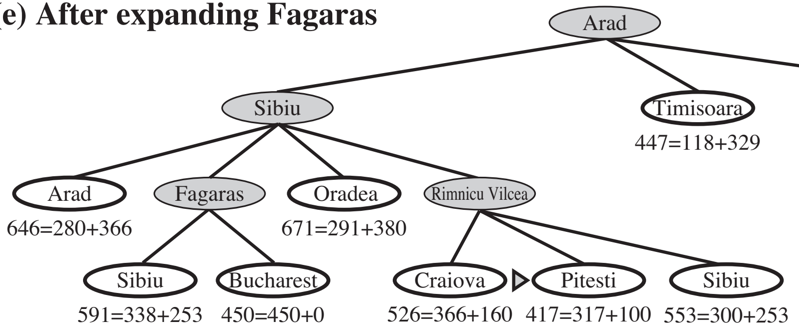


A* Search Example

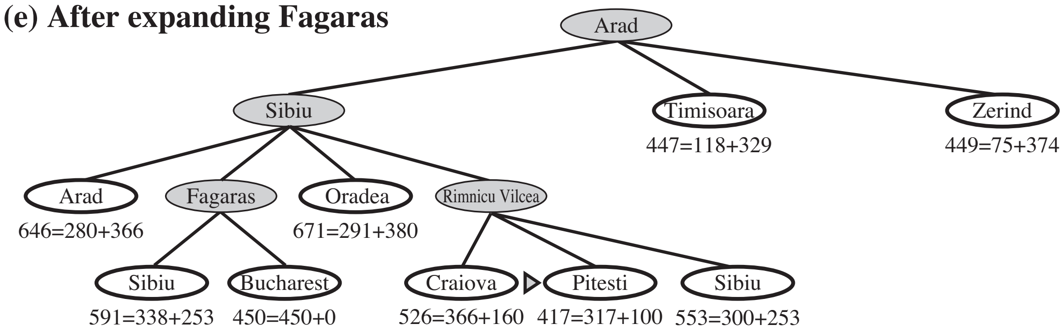


A* Search Example

(e) After expanding Fagaras

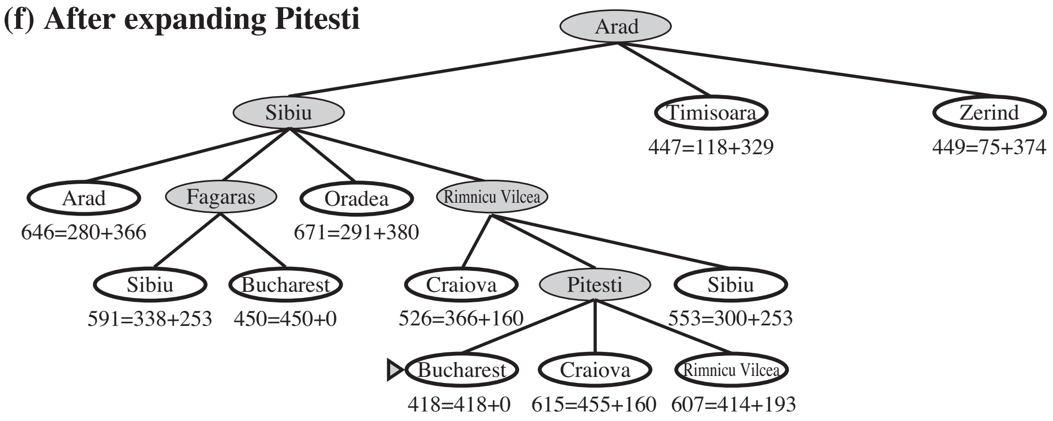


A* Search Example



Should we stop here?

A* Search Example



A* Search

- In the case of tree search, A* is optimal **only** if $h(n)$ is an **admissible heuristic**
  - A heuristic is admissible, or "optimistic" if it **never overestimates** the cost to reach the goal
- Using such a heuristic means that $f(n)$ never overestimates the cost of reaching the goal
- The straight line distance is admissible. Why?

Iterative Deepening A* (IDA*)

- Extends the idea of Depth-First Iterative-Deepening (DFID) Search by including the heuristic estimate
- Instead of a **depth limit**, uses an **upper bound** U on the f-value
  - U is a **current threshold**, such that no node with $f > U$ is expanded
  - During search, computes an upper bound U' for the next iteration
- Like DFID in terms of overheads and advantages
  - Limits memory usage by going “depth-first” in the heuristic
  - Overhead dependent on the diversity of f-values

```
procedure IDA*-driver(problem $p$, heuristic $h$)
    global $U' \leftarrow h(p.initial)$                    ▷ Initialize Threshold
    $bestPath \leftarrow \emptyset$                        ▷ Initialize Solution Path
    while $bestPath = \emptyset$ and $U' \neq \infty$ do   ▷ Goal not found, nodes left
        $U \leftarrow U'$                                  ▷ Reset global threshold
        $U' \leftarrow \infty$                             ▷ Initialize new global threshold
        $bestPath \leftarrow$ IDA*$(p.initial, p, 0, U, h)$
    return $bestPath$                                      ▷ Terminate with solution path
function IDA*(node $u$, problem $p$, current cost $g$, bound $U$, heuristic $h$)
    if P.goalTest$(u)$ then return GetPath$(u)$            ▷ Terminate Search
    for all $a \in$ p.actions$(u)$ do                      ▷ For all successors
        $v \leftarrow$ a.result$(u)$
        if $g + a.cost + h(v) > U$ then                    ▷ Cost exceeds old bound
            if $g + a.cost + h(v) < U'$ then               ▷ Cost smaller than new bound
                $U' \leftarrow g + a.cost + h(v)$          ▷ Update new bound
        else                                               ▷ $f$-value below current threshold
            $\pi \leftarrow$ IDA*$(v, p, g + a.cost, U, h)$
            if $\pi \neq \emptyset$ then return $\pi$      ▷ Solution found
    return $\emptyset$                                     ▷ No solution exists
```

Weighted A* Search

- Explores states by increasing weighted-plan-cost estimate $g + W * h$
- Exactly like A*, except for the weight parameter

```
function wAStarSearch(problem $p$, heuristic $h$, weight $w$)
    $closed \leftarrow \{\}$
    $frontier$.add$(newNode(p.initial), 0)$
    loop
        if $frontier$ is empty then
            return $fail$
        $n \leftarrow frontier$.get
        $closed$.add$(n.state)$
        if p.goalTest$(n.state)$ then
            return getPath$(n)$
        for all $a \in p$.actions$(n)$ do
            $n' \leftarrow a$.result$(n.state)$
            if $n'.state \notin closed$ then
                $v \leftarrow n'.cost + w * h(n'.state)$
                $frontier$.add$(n', v)$
```

```
function wAStarSearch(problem $p$, heuristic $h$, weight $w$)
    $closed \leftarrow \{\}$
    $frontier$.add($newNode(p.initial), 0$)
    loop
        if $frontier$ is empty then
            return $fail$
        $n \leftarrow frontier$.get
        $closed$.add($n.state$)
        if p.goalTest($n.state$) then
            return getPath($n$)
        for all $a \in p$.actions($n$) do
            $n' \leftarrow a$.result($n.state$)
            if $n'.state \notin closed$ then
                $v \leftarrow n'.cost + w * h(n'.state)$    ▷ Full cost estimate, weighed heuristic
                $frontier$.add($n', v$)
```

Weighted A* Search

- Explores states by increasing weighted-plan-cost estimate $g + W * h$
- The weight $W \in \mathbb{R}_0^+$ is an algorithm parameter:
  - For $W = 0$ how does weighted A* behave?
  - For $W = 1$ how does weighted A* behave?
  - For $W = 10^{100}$ how does weighted A* behave?
- Properties:
  - For $W > 1$ weighted A* is **bounded suboptimal**

Weighted A* Search

- Explores states by increasing weighted-plan-cost estimate $g + W * h$
- The weight $W \in \mathbb{R}_0^+$ is an algorithm parameter:
  - For $W = 0$ how does weighted A* behave? UCS
  - For $W = 1$ how does weighted A* behave?
  - For $W = 10^{100}$ how does weighted A* behave?
- Properties:
  - For $W > 1$ weighted A* is **bounded suboptimal**

Weighted A* Search

- Explores states by increasing weighted-plan-cost estimate $g + W * h$
- The weight $W \in \mathbb{R}_0^+$ is an algorithm parameter:
  - For $W = 0$ how does weighted A* behave? UCS
  - For $W = 1$ how does weighted A* behave? A*
  - For $W = 10^{100}$ how does weighted A* behave?
- Properties:
  - For $W > 1$ weighted A* is **bounded suboptimal**

Weighted A* Search

- Explores states by increasing weighted-plan-cost estimate $g + W * h$
- The weight $W \in \mathbb{R}_0^+$ is an algorithm parameter:
  - For $W = 0$ how does weighted A* behave? UCS
  - For $W = 1$ how does weighted A* behave? A*
  - For $W = 10^{100}$ how does weighted A* behave?GBFS
- Properties:
  - For $W > 1$ weighted A* is **bounded suboptimal**

Outline

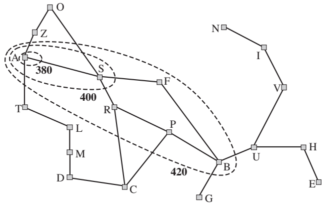
Admissible Heuristics

- Tree search with A* is optimal only if the heuristic used is admissible
  - Assume that there is suboptimal goal node G in the frontier, and let the cost of the optimal solution be C^*. Now, $f(G) = g(G) + h(G)$
  - Since $h(G) = 0$ (as h is admissible), $g(G) > C^*$
  - Now consider a different node on an optimal solution path. If h does not overestimate, then $f(n) = g(n) + h(n) \leq C^*$
  - So $f(n) \leq C^* < f(G)$ so G will not be expanded, and therefore A^* must return an optimal solution

A^* in a Graph

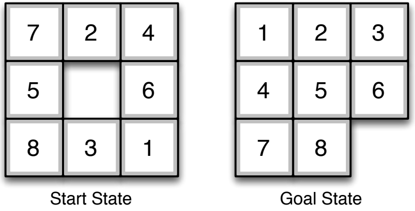
- Optimality of admissible heuristics in a graph search requires either
  - Always discarding the more expensive of any two paths to the same node; or
  - Ensuring that the optimal path to any repeated state is always the first one followed
- The second property holds if h is **consistent** (a.k.a **monotonic**): if for every node n and successor n' generated by a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'
  $h(n) \leq c(n, a, n') + h(n')$
- If h is consistent, then the values of f along any path are nondecreasing

Contours



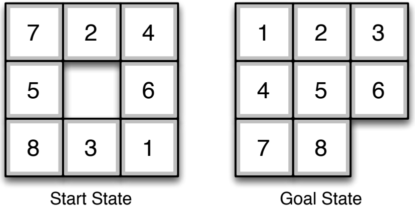
- Accurate heuristic functions cause the contour to stretch towards the optimal path more narrowly
- A* search is very good, but the number of nodes within the goal contour search space is still exponential in the length of the solution (unless the heuristic is very very good)
- There are other search techniques which use much less memory at the cost of increased computation time.
- For many problems, the optimality requirement must be dropped.

Heuristic Functions



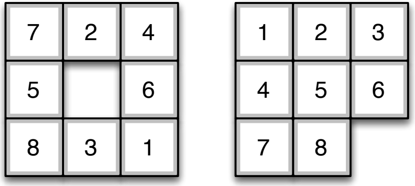
- Branching factor:
- Average depth:
- Total number of states:

Heuristic Functions



- Branching factor: 3
- Average depth: 22
- Total number of states: 3^{22}

Heuristic Functions

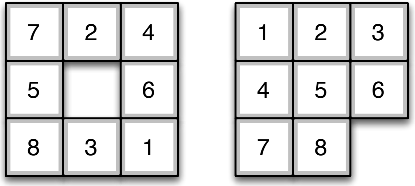


Start State Goal State

- Heuristics:

- Are they admissible?

Heuristic Functions



Start State   Goal State

- Heuristics:
  - The number of misplaced tiles – **hamming distance** ($h_1 = 6$)
  - The sum of distances of the tiles from their goal positions – **manhattan distance** ($h_2 = 14$)
- Are they admissible?

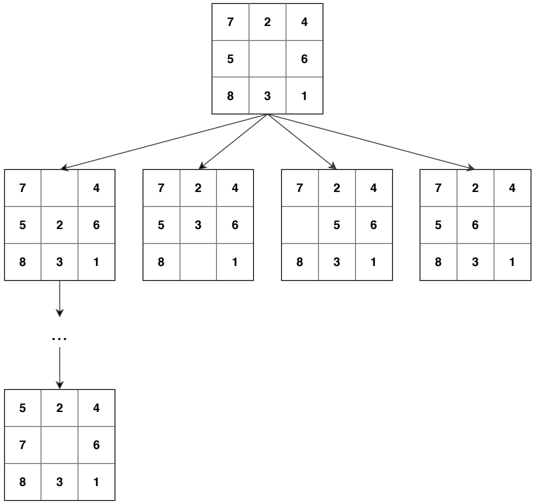
Domination

- h_2 is always greater than h_1
  - It therefore dominates h_1 (it is a tighter bound on the actual cost)
  - And is more efficient
- How can we invent good heuristics?
- Typically, by relaxing the problem.

Generating Heuristics

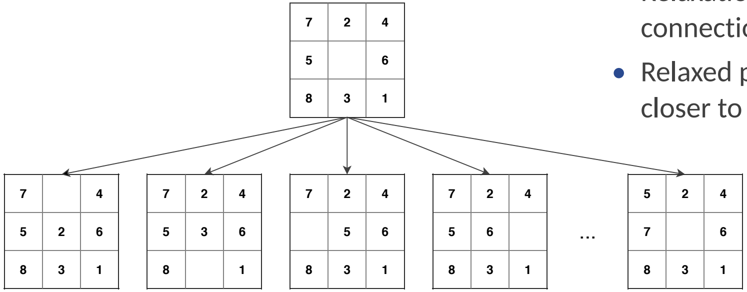
- Rule: a tile can move from square A to square B if
  - A is horizontally or vertically adjacent to B; and
  - B is blank
- We can generate relaxed problems by removing one or both of the conditions
  - Removing 2nd condition leads to manhattan distance.
  - Removing both conditions leads to h_1
- There are other ways to obtain heuristics (e.g., learning)

How to Relax



- Relaxation effectively creates new connections in the state-space graph

How to Relax



- Relaxation effectively creates new connections in the state-space graph
- Relaxed problem includes solutions closer to the root

Heuristic $\neq$ Search

Search algorithms **are not** the same as the heuristics that use them!

- Informed search algorithms use them
- Optimal search algorithms depend on admissible heuristics (A^*)

Heuristic Search Summary

- We've investigated uninformed and informed search
  - DFS, BFS, UCS
  - Greedy, A*
- There are many other types of searches
  - Reduce memory usage
  - Deal with continuous environments
  - Deal with interleaving search and acting, etc.

Any Questions.