Informed Search

Czech Technical University in Prague, Faculty of Electrical Engineering

ZUI 2011, course 2

Search Problems

- State Space S - stavový prostor
- Initial state s_0 – počáteční stav
- Successor function: $\forall s \in S: successor(s) \rightarrow \{s_1, s_2, \dots, s_n\}$
- Goal Test $s \in S: goal(s) \rightarrow T \text{ or } F$
- Arc cost $s \in S: g(s) \rightarrow \boldsymbol{R}$
- Heuristic $s \in S: h(s) \rightarrow \boldsymbol{R}$

Informed Search Problems

- State Space S - stavový prostor
- Initial state s_0 – počáteční stav
- Successor function: $\forall s \in S: successor(s) \rightarrow \{s_1, s_2, \ldots, s_n\}$
- Goal Test $s \in S: goal(s) \rightarrow T\ or\ F$
- **Arc cost** $\boldsymbol{s \in S: g(s) \rightarrow R}$
- **Heuristic** $\boldsymbol{s \in S: h(s) \rightarrow R}$

Note on algorithm properties

- **Optimal** – The algorithm returns best solution.
- **Complete** – if solution exists, the algorithm finds a solution. If not, the algorithm reports that no solution exists.
- **Sound** – Complete and Optimal algorithm
- **Admissible** – Optimal

General Search Algorithm Template

1. If GOAL?(initial-state) then return initial-state
2. INSERT(initial-node,FRINGE)
3. Repeat:
   a. If empty(FRINGE) then return failure
   b. N ← REMOVE(FRINGE)
   c. s ← STATE(N)
   d. For every state s' in SUCCESSORS(s)
      i. Create a new node N' as a child of N
      ii. If GOAL?(s') then return path or goal state
      iii. INSERT(N',FRINGE)

Heuristic Search Algorithm Template

1. If GOAL?(initial-state) then return initial-state
2. INSERT(initial-node,FRINGE)
3. Repeat:
   a. If empty(FRINGE) then return failure
   b. N ← REMOVE(FRINGE)
   c. s ← STATE(N)
   d. If GOAL?(s) then return path or goal state
   e. For every state s' in SUCCESSORS(s)
      i. Create a new node N' as a child of N
      ii. INSERT(N',FRINGE)

General Search Algorithm Template

1. If GOAL?(initial-state) then return initial-state
2. INSERT(initial-node,FRINGE)
3. Repeat:
   a. If empty(FRINGE) then return fail
   b. N ← REMOVE(FRINGE)
   c. s ← STATE(N)
   d. If GOAL?(s) then return path or goal state
   e. For every state s' in SUCCESSORS(s)
      i. Create a new node N' as a child of N
      ii. INSERT(N',FRINGE)

Goal test

A* Search

- **Idea:** Avoid extending paths that seem to be expensive
- Best-First search
- Evaluation function f(n) for each state/node

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

COST

HEURISTIC

- g(n): cost to reach the node n
- h(n): **estimated** cost to get from node n to goal

A* continued

1. If GOAL?(initial-state) then return initial-state
2. INSERT(initial-node,FRINGE)
3. Repeat:
   a. If empty(FRINGE) then return failure
   b. N ← REMOVE(FRINGE)
   c. s ← STATE(N)
   d. If GOAL?(s') then return path or goal state
   e. For every state s' in SUCCESSORS(s)
      i. Create a new node N' as a child of N
      ii. INSERT(N',FRINGE)

A* continued

1. If GOAL?(initial-state) then return initial-state
2.
3.

Fringe: Sorted List – lowest values first

N – inserted according to its value

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

c. s ← STATE(N)

d. If GOAL?(s') then return path or goal state

e. For every state s' in SUCCESSORS(s)

   i. Create a new node N' as a child of N

   ii. INSERT(N',FRINGE)

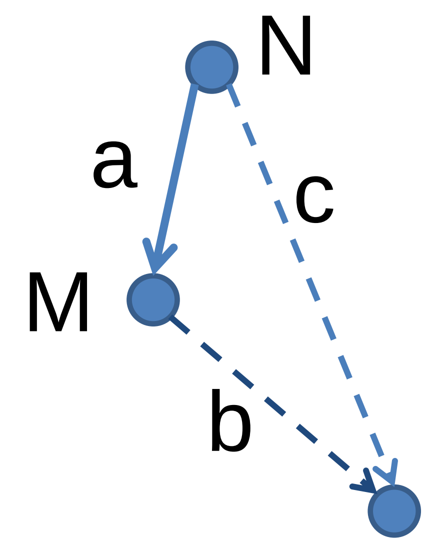
H(N) – Heuristic function

- We know the cost to the node g(n) – nothing to tune here
- We don’t know the exact cost from n to goal h(n) – if we knew, no need to search – **estimate it!**

- H(N) – **admissible** and **consistent** heuristic
- Admissible = optimistic – it never overestimates the cost to the goal
- Consistent = Triangle inequality is valid

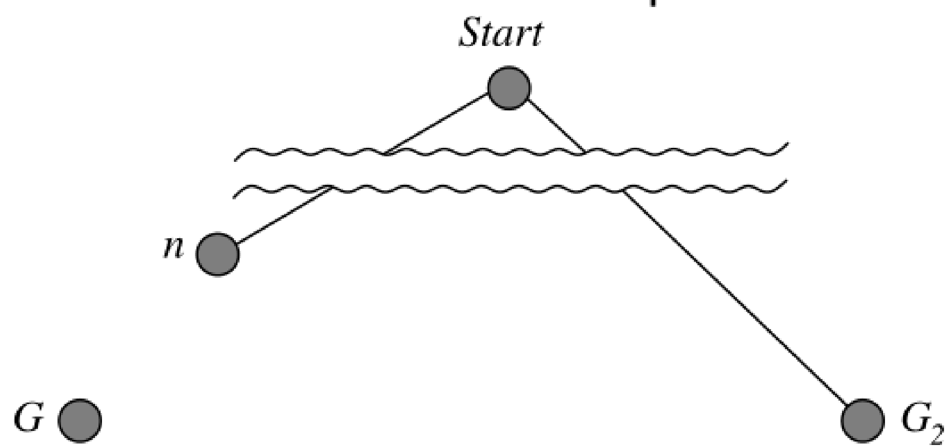
$$-a + b \geq c$$

$$-g(M) + h(M) \geq h(N)$$



Optimality of A* (standard proof)

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.



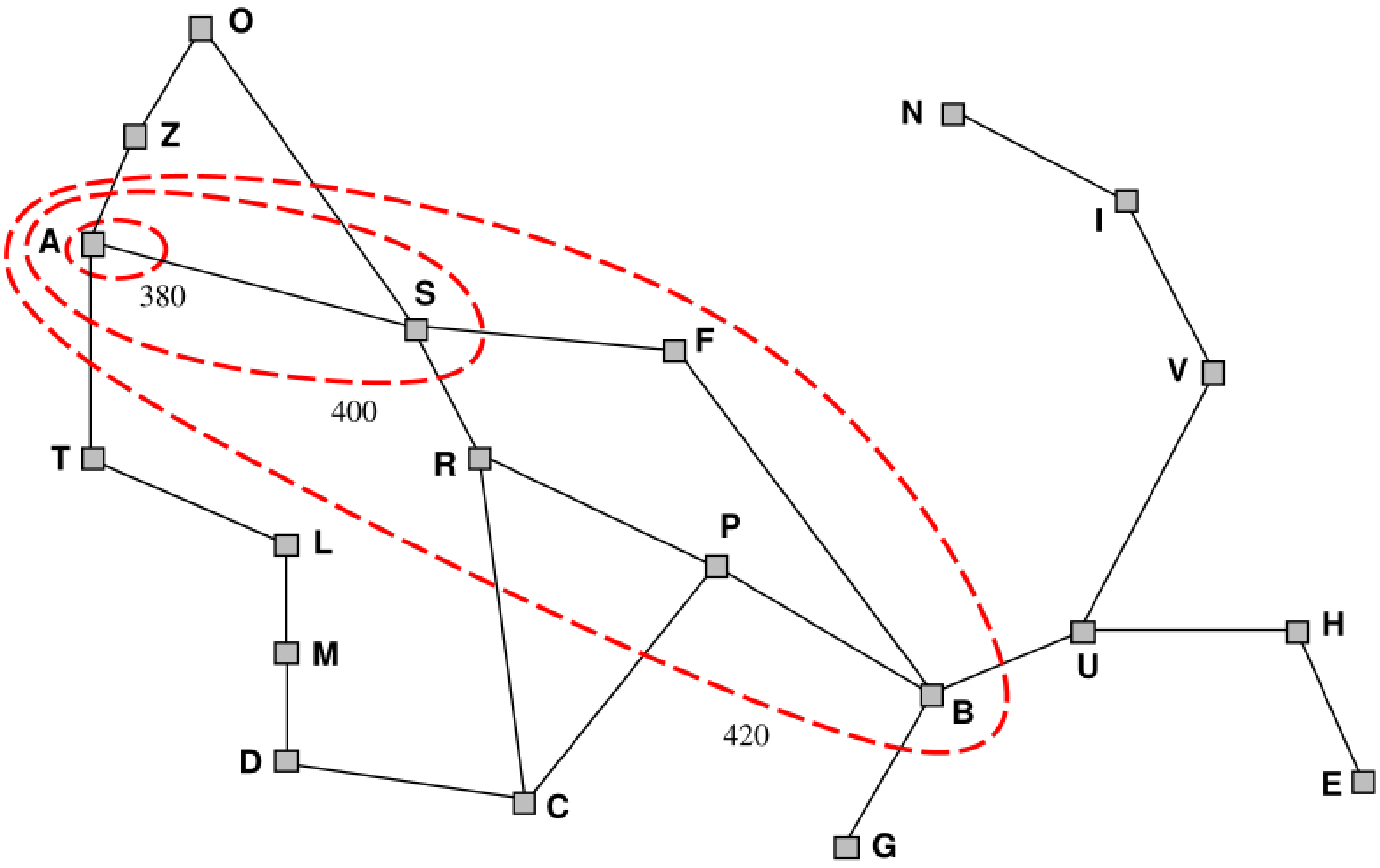
$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible} \end{aligned}$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

Optimality of A* (more useful)

Lemma: A* expands nodes in order of increasing f value*

Gradually adds "f-contours" of nodes (cf. breadth-first adds layers)
Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$



Properties of A*

- Complete, unless there are infinitely many nodes with $f(n) \leq f(G)$
- Runtime complexity – exponential in [relative error in h]
- Space complexity – keeps all nodes in memory
- Optimal

- A* expands all nodes with $f(n) < C^*, g(G) = C^*$
- A* expands some nodes with $f(n) = C^*$
- A* expands no nodes with $f(n) > C^*$

Escaping the World Trade Center

- Imagine a huge skyscraper with several elevators. As the input you have:
- set of elevators, where for each you have:
- - range of the floors that this elevator is operating in
- - how many floors does this elevator skip (e.g. an elevator can stop only on every second floor, or every fifth floor, etc.)
- - speed (time in seconds to go up/down one floor)
- - starting position (number of the floor)

Escaping the World Trade Center

- Let us assume, that transfer from one elevator to another one takes the same time (given as input - t).
- You are starting in k^{th} floor and you want to find the quickest way to the ground floor.
- You can assume that you are alone in the building and elevators do not run by themselves.

1. **What are the states?**
2. **What is the initial state and the goal state?**
3. **What is the cost function?**
4. **What are possible heuristics?**