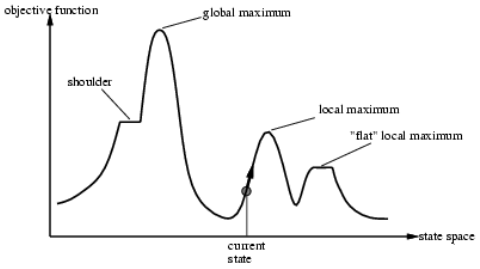
Hill-climbing search

- Problem: depending on initial state, can get stuck in local maxima,…etc



Problems with hill climbing

1. Local maximum problem: there is a peak, but it is lower than the highest peak in the whole space.
2. The plateau problem: all local moves are equally unpromising, and all peaks seem far away.
3. The ridge problem: almost every move takes us down.

Solution:

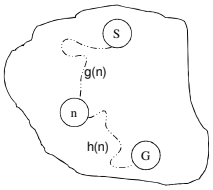
Random-restart hill climbing is a series of hill-climbing searches with a randomly selected start node whenever the current search gets stuck.

Algorithm A*

- One of the most important advances in AI search algs.
- Idea: avoid expanding paths that are already expensive

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

- *g(n)* = least cost path to n from S found so far
- *h(n)* = estimated cost to goal from n
- *f(n)* = estimated total cost of path through n to goal



The A* procedure

Hill-climbing (and its improved versions) may miss an optimal solution. Here is a search method that ensures **optimality** of the solution.

The algorithm

keep a list of partial paths (initially root to root, length 0);
repeat
succeed if the first path P reaches the goal node;
otherwise remove path P from the list;
extend P in all possible ways, add new paths to the list;
sort the list by **the sum of two values:** the real cost of P till now, and an estimate of the remaining distance;
prune the list by leaving only the shortest path for each node reached so far;
until
success or the list of paths becomes empty;

The A* procedure

A heuristic that never overestimates is also called **optimistic** or **admissible**.

We consider three functions with values ≥ 0:

- g(n) is the actual cost of reaching node n,
- h*(n) is the actual *unknown* remaining cost,
- h(n) is the optimistic estimate of h(n).

Admissible heuristics

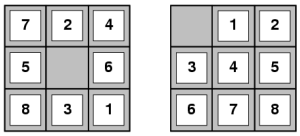
- A heuristic $h(n)$ is admissible if for every node n, $h(n) \le h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Theorem: If $h(n)$ is admissible, A* using is optimal

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



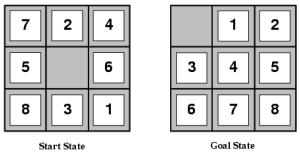
- $h_1(S)$ = ?
- $h_2(S)$ = ?

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



- $h_1(S)$ = ? 8
- $h_2(S)$ = ? 3+1+2+2+2+3+3+2 = 18

Admissible heuristics

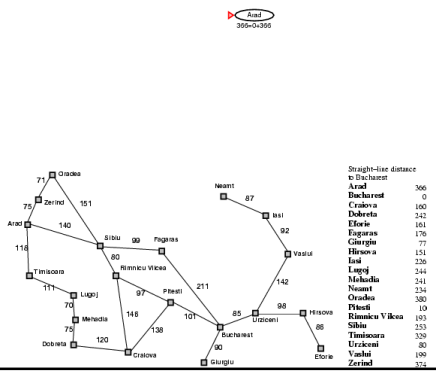
- If $h_2(n) >= h_1(n)$ for all n, both are admissible
- Then h_2 dominates h_1 and is usually better for search

Typical Costs

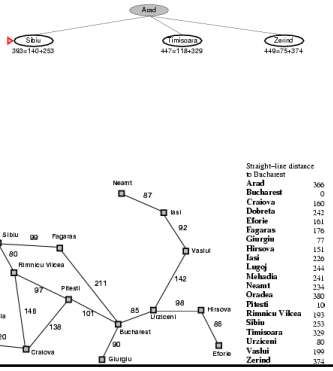
- d = 14 IDS = 3,473,941 nodes
  - A*(h_1) = 539 nodes
  - A*(h_2) = 113 nodes
- d = 24 IDS ~ 54,000,000,000 nodes
  - A(h_1) = 39,135 nodes
  - A(h_2) = 1,641 nodes

Remark: Given h_1 and h_2 any two admissible functions then $h(n) = \max\{h_1(n), h_2(n)\}$ is also admissible

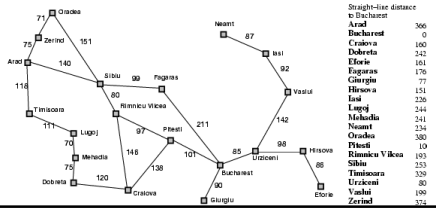
A* search example



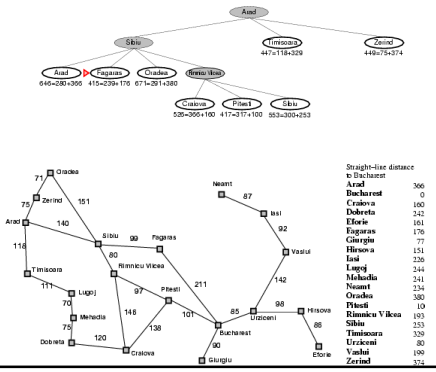
A* search example



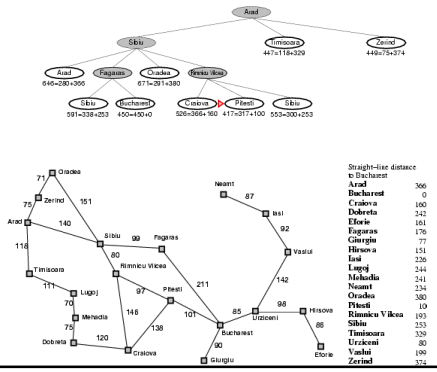
A* search example



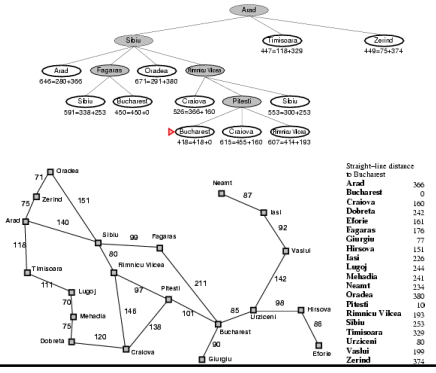
A* search example



A* search example



A* search example



Properties of A*

- Complete?

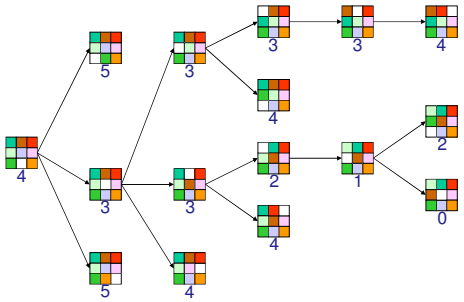
Yes (unless there are infinitely many)

- Time/Space?
- Exponential mostly
- Optimal?

Yes

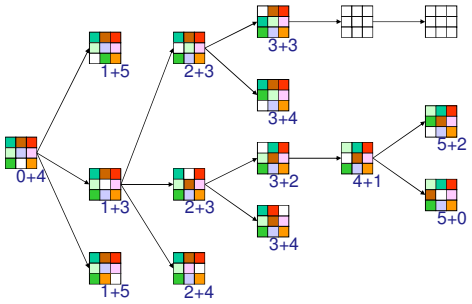
8-Puzzle

f(N) = h(N) = number of misplaced tiles



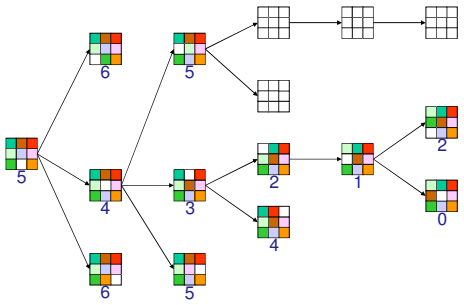
8-Puzzle

f(N) = g(N) + h(N)
with h(N) = number of misplaced tiles



8-Puzzle

f(N) = h(N) = Σ distances of tiles to goal



Local Search Algorithms

- In many optimization problems, ***path*** is irrelevant
- the goal state itself is the solution
- Ex: The 8-queen problem, the final configuration of the queens is the important not the order they were put
- Operates using only single current state, rather than multiple paths.
- Find Optimal Configuration (satisfies the constraints)
- Use ***iterative improvement algorithms***
- A **Complete** local search algorithm finds a goal if exists
- An **Optimal** algorithm finds the global minimum or maximum