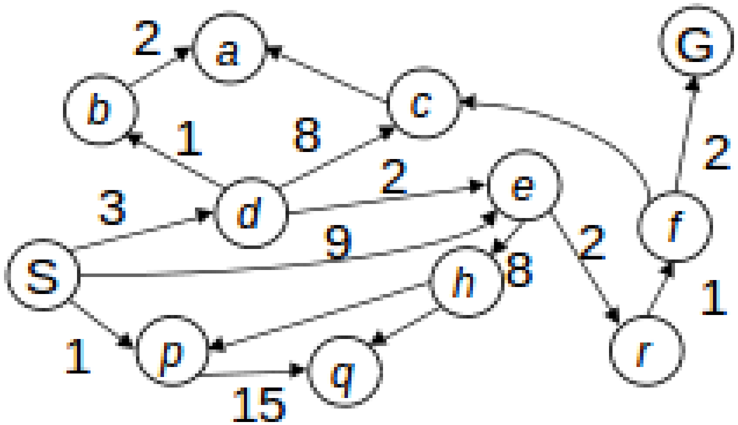
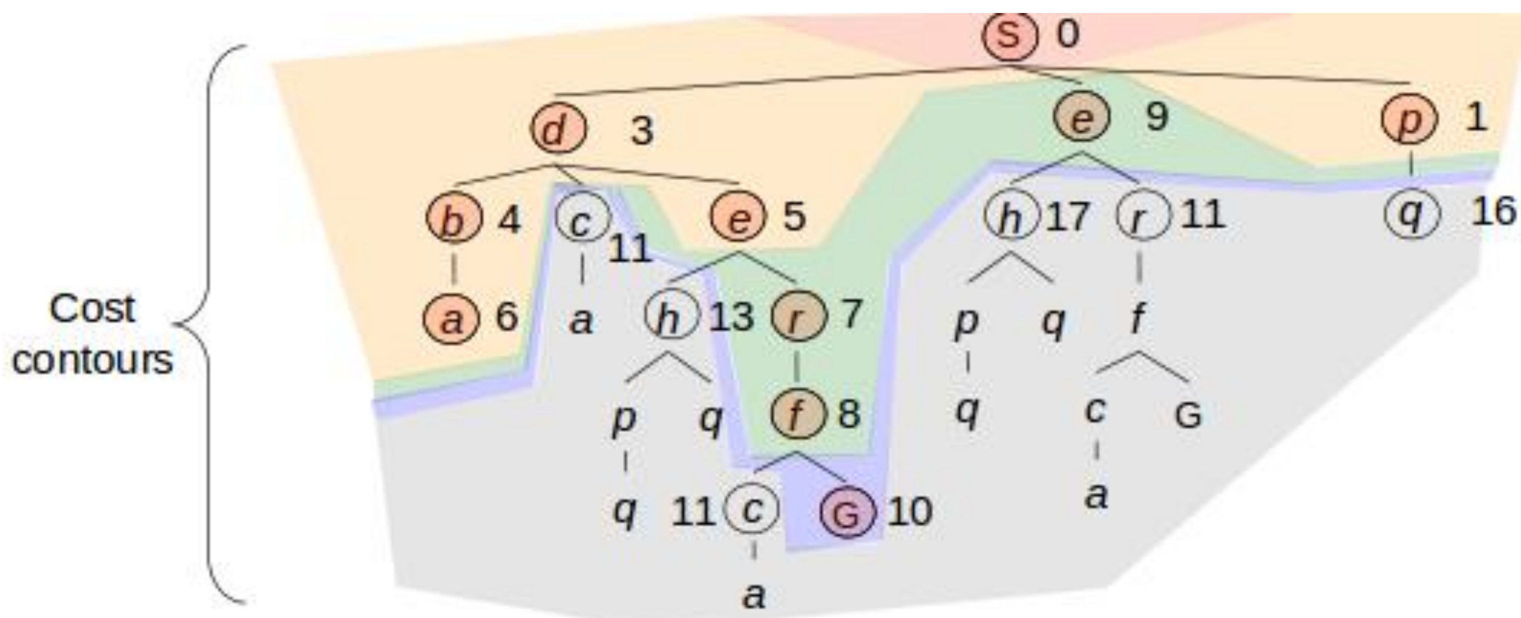
Uniform Cost Search



- Strategy: Expand cheapest node first.

- Data structure: ?



Uniform Cost Search: Algorithm

- Insert the root node into the priority queue
- *Repeat while the queue is not empty:*

  Remove the element with the highest priority

  If the removed node is the destination,

  print total cost and stop the algorithm

  Else,

  enqueue all the children of the current node to the priority queue, with their cumulative cost from the root as priority.

Heuristic Search

Outline

- Best-first search
- Greedy best-first search
- A^* search
- Heuristics
- Local search algorithms
- Hill-climbing search

Heuristic Search

- Heuristics are criteria for deciding which among several alternatives be the most effective in order to achieve some goal.
- Heuristic is a technique that
  - improves the efficiency of a search process possibly by sacrificing claims of systematicity and completeness.
  - It no longer guarantees to find the best answer but almost always finds a very good answer.

Heuristic Search – Contd..

- Using good heuristics, we can hope to get good solution to hard problems in less than exponential time.
- There are **general-purpose** heuristics that are useful in a wide variety of problem domains.
- We can also construct **special purpose** heuristics, which are domain specific.

General Purpose Heuristics

- A general-purpose heuristics for combinatorial problem is
  - **Nearest neighbor algorithms** which works by selecting the locally superior alternative.
- In many AI problems,
  - it is often hard to measure precisely the goodness of a particular solution.
  - But still it is important to keep performance question in mind while designing algorithm.

Contd…

- For real world problems,
  - it is often useful to introduce heuristics based on relatively unstructured knowledge.
  - It is impossible to define this knowledge in such a way that mathematical analysis can be performed.
- In AI approaches,
  - behavior of algorithms are analyzed by running them on computer as contrast to analyzing algorithm mathematically.

Best-first search

- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability"

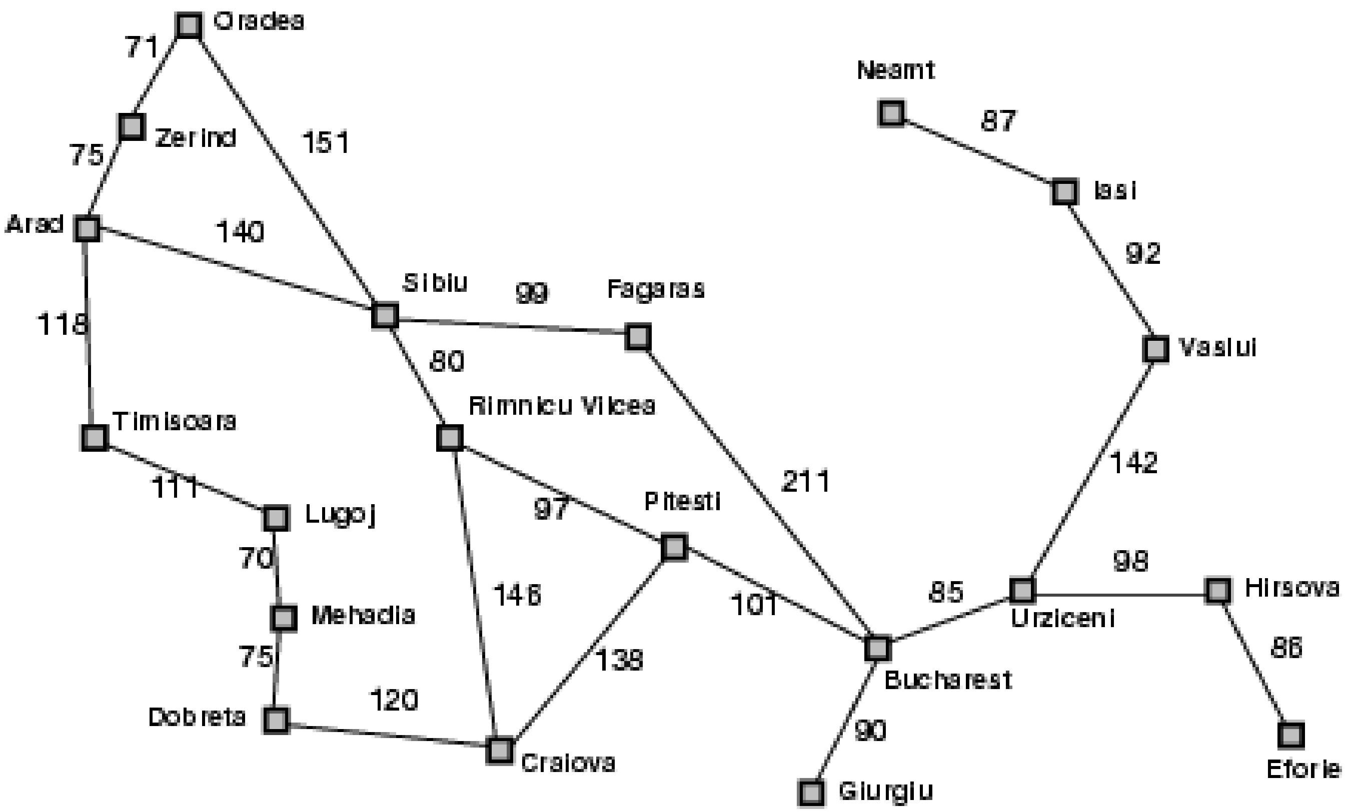
  → Expand most desirable unexpanded node

- Implementation:

  Order the nodes in fringe in decreasing order of desirability

- Special cases:
  - greedy best-first search
  - A^* search

Romania with step costs in km



Straight–line distance to Bucharest

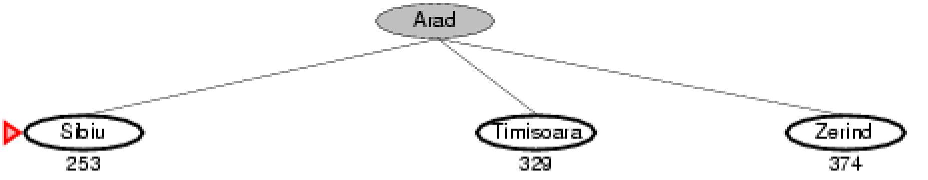
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

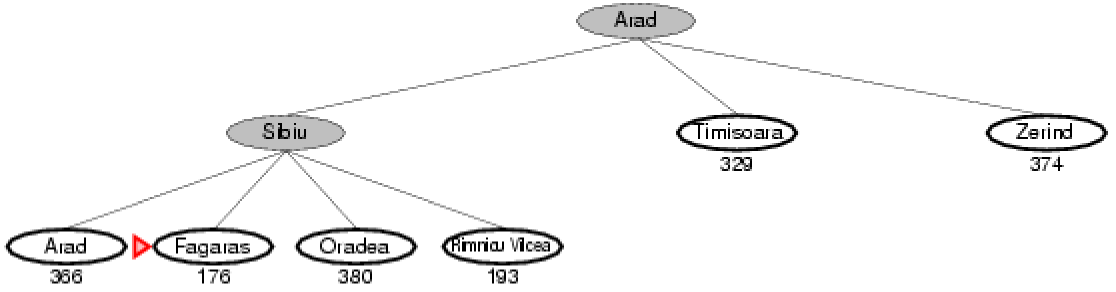
- Evaluation function $f(n) = h(n)$ (heuristic)
- = estimate of cost from *n* to *goal*
- e.g., $h_{SLD}(n)$ = straight-line distance from *n* to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal

Greedy best-first search example

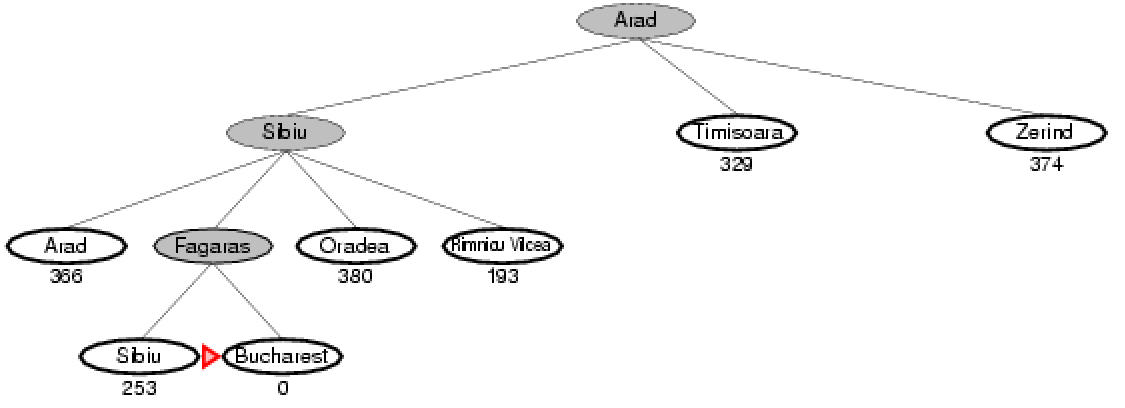
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



Properties of greedy best-first search

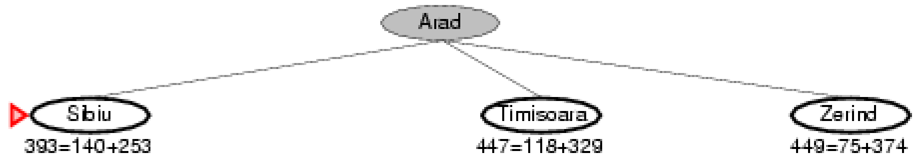
- Complete?
- No – can get stuck in loops, e.g., Iasi → Neamt → Iasi → Neamt →
- Time?
- *O(b^m)*, but a good heuristic can give dramatic improvement
- Space?
- *O(b^m)* -- keeps all nodes in memory
- Optimal?
- N

A^* search

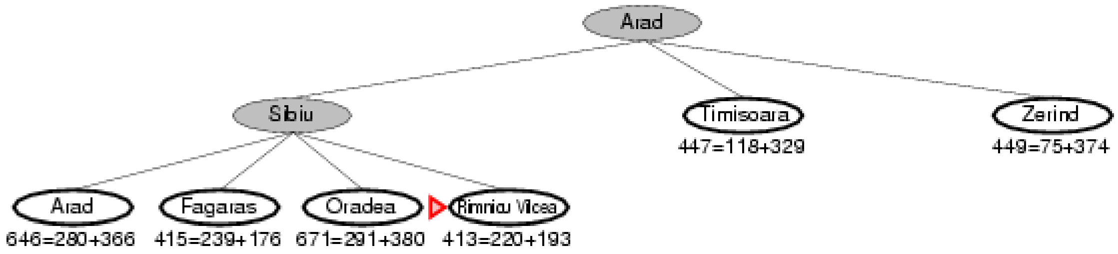
- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal

A^* search example

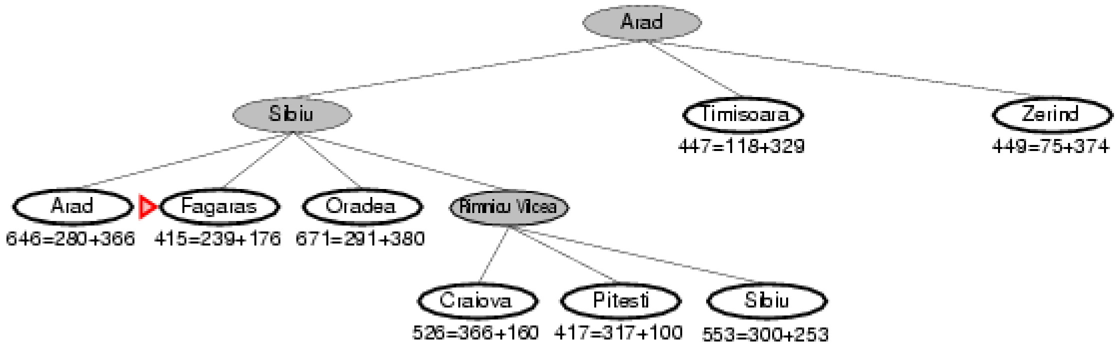
A* search example



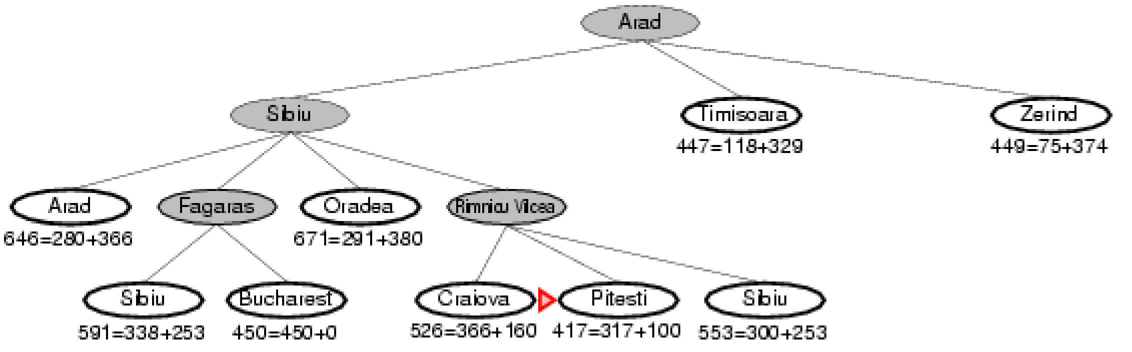
A* search example



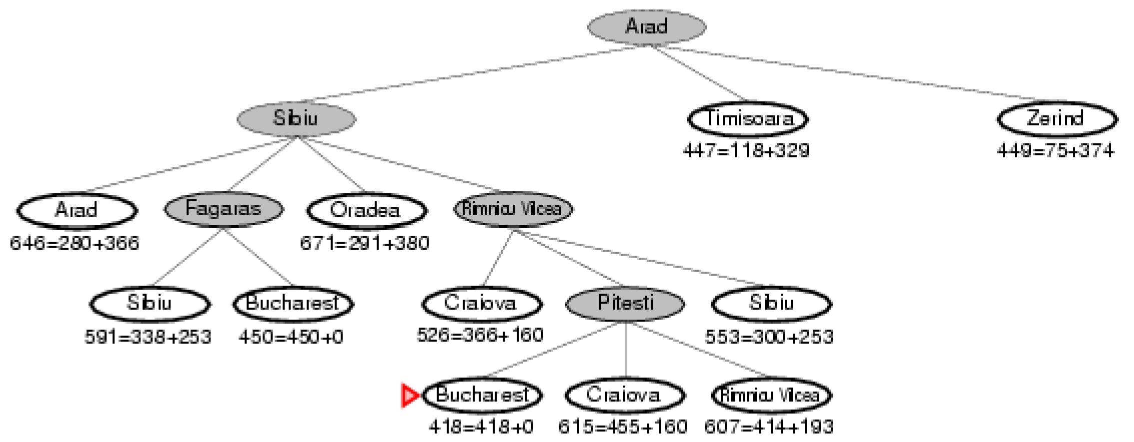
A* search example



A* search example



A^* search example



Admissible heuristics

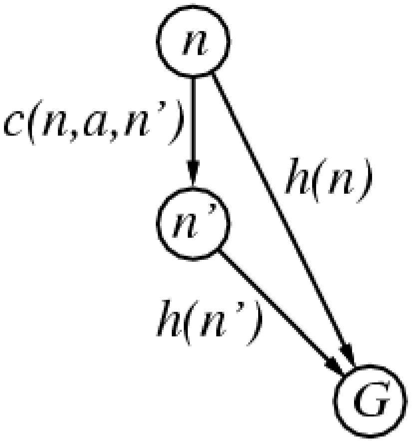
- A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.

- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

Consistent heuristics

- A heuristic is consistent if for every node *n*, every successor *n'* of *n* generated by any action *a*,

$$h(n) \leq c(n,a,n') + h(n')$$



Properties of A*

- Complete?
- Yes (unless there are infinitely many nodes with f ≤ *f(G)*)
- Time?
- Exponential
- Space?
- Keeps all nodes in memory
- Optimal?
- Yes

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible)
- then h_2 dominates h_1
- h_2 is better for search

- Typical search costs (average number of nodes expanded):

- *d=12* IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes
- *d=24* IDS = too many nodes
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes
-

Relaxed problems

- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution

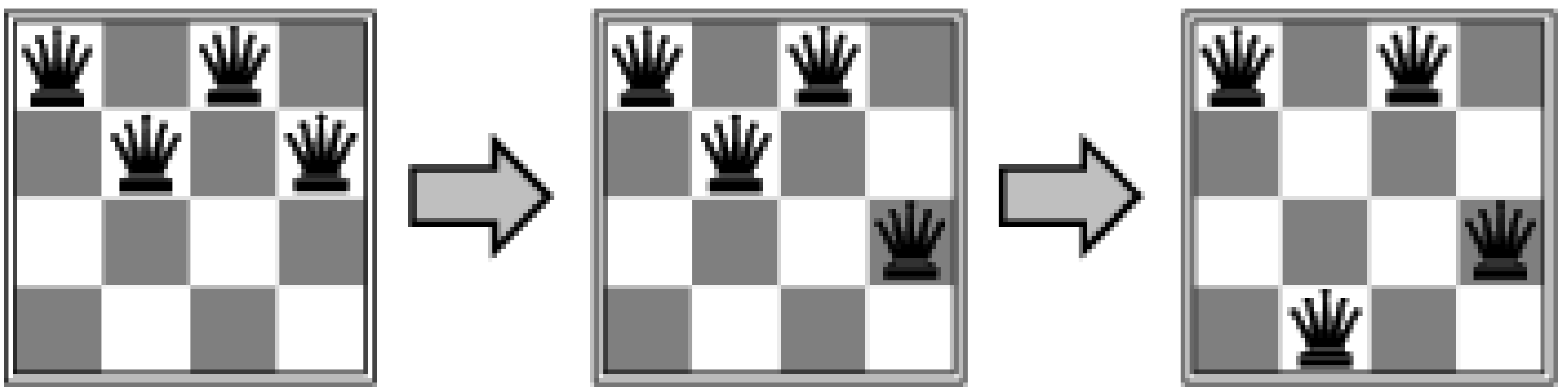
Local search algorithms

- In many optimization problems, the path to the goal is irrelevant; the goal state itself is the solution

- State space = set of "complete" configurations
- Find configuration satisfying constraints, e.g., n-queens

- In such cases, we can use local search algorithms
- keep a single "current" state, try to improve it

Example: *n*-queens

- Put *n* queens on an *n* × *n* board with no two queens on the same row, column, or diagonal
- 

Hill Climbing- Algorithm

Generate and Test *Algorithm*

Start

- Generate a possible solution
- Test to see, if it is goal.
- If not go to start else quit

End

Hill Climbing- **(Variant of generate and test strategy)**

- Search efficiency may be improved if there is some way of ordering the choices so that the most promising node is explored first.
- Moving through a tree of paths, hill climbing proceeds
  - in depth-first order but the choices are ordered according to some heuristic value (i.e, measure of remaining cost from current to goal state).

Example of heuristic function

- Straight line (as the crow flies) distance between two cities may be a heuristic measure of remaining distance in traveling salesman problem .

Hill climbing : Algorithm

- Store initially, the root node in a OPEN list (maintained as stack) ;   Found = false;
- While (OPEN $\neq$ empty and Found = false)   Do

{

  - Remove the top element from OPEN list and call it NODE;
  - If NODE is the goal node, then **Found = true** else find SUCCs, of NODE, if any, and **sort SUCCs** by estimated cost from NODE to goal state and add them to the front of OPEN list.

} /* end while */

- If **Found = true** then return **Yes** otherwise return **No**
- Stop

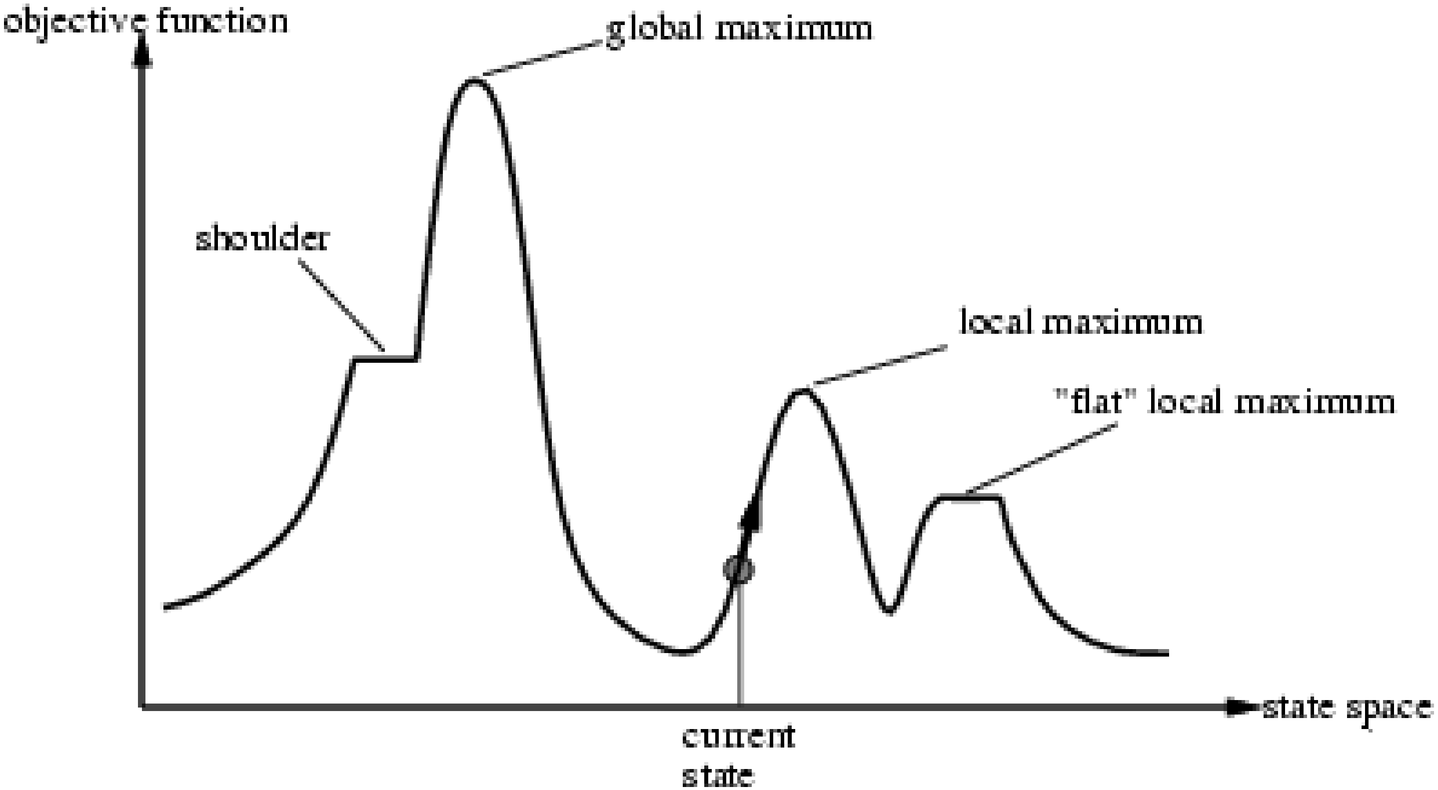
Problems in hill climbing

- There might be a position that is not a solution but from there no move improves situations?
- This will happen if we have reached a *Local maximum*, a *plateau* or a *ridge.*
  - **Local maximum:** It is a state that is better than all its neighbors but is not better than some other states farther away. All moves appear to be worse.
    - *Solution to this is to backtrack to some earlier state and try going in different direction.*

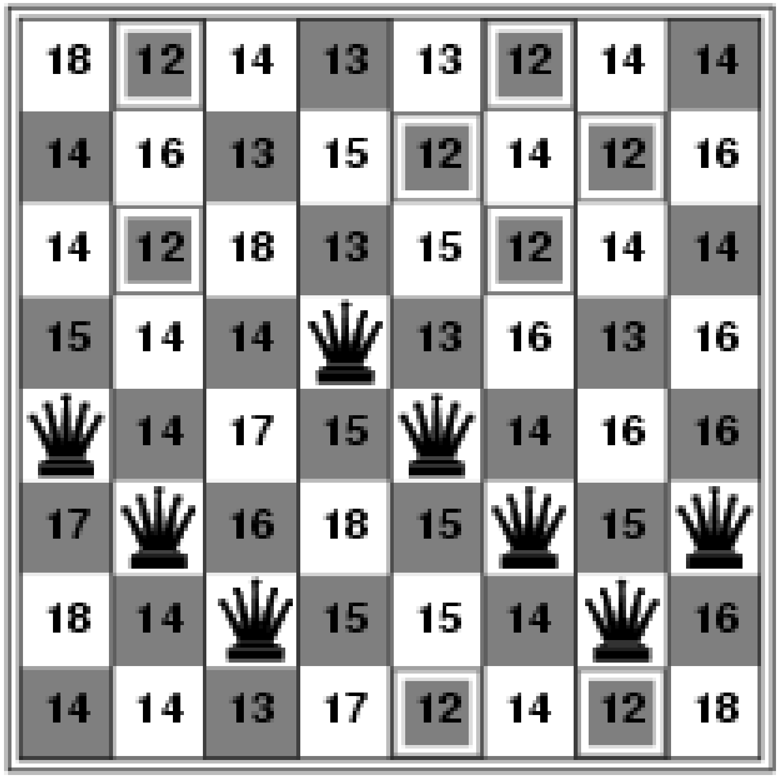
Contd…

- **Plateau:** It is a flat area of the search space in which, a whole set of neighboring states have the same value. It is not possible to determine the best direction.
    - *Here make a big jump to some direction and try to get to new section of the search space.*
    - *Here apply two or more rules before doing the test i.e., moving in several directions at once.*

Hill-climbing search

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

Hill-climbing search: 8-queens problem



- *h* = number of pairs of queens that are attacking each other, either directly or indirectly
- *h = 17* for the above state
-

Hill-climbing search: 8-queens problem