CISS450: Artificial Intelligence
Lecture 6: Local Search

Yihsiang Liow

October 28, 2024

Table of contents I

Agenda

- Local search algorithms
- Optimization problems

Local Search I

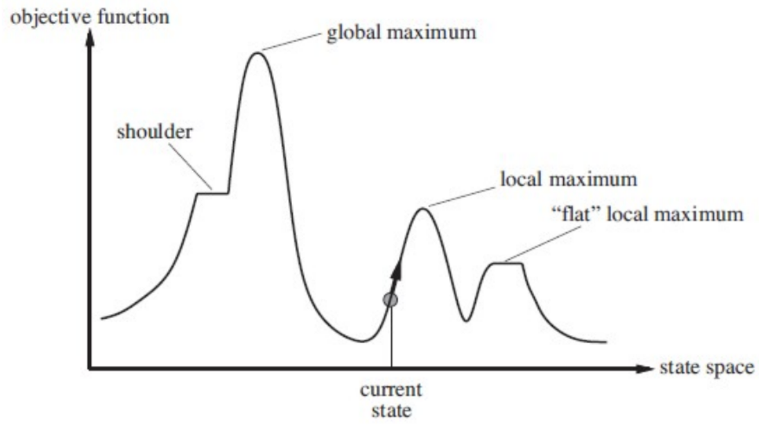
Local Search:

- Keeps track of only one state
  - Use very little memory
- Move from one state to another until best is found
- Usually path of computation is not important – the final state is.
- Not all states are considered
  - not as systematic as other algorithms we've seen so far

Local Search I

- For optimization problems, there is an objective function that describes the "quality" of a state. So in moving from one state to another, choice of new state is based on objective function.
  - For some cases, you want to maximize the objective function. For other cases, the goal is to minimize the objective function.

Local Search I



Hill Climbing I

- The idea is to continually search for a state by going to the next state with a highest value for a function f. If there's more than one, randomly choose one of the best.
- If successor with highest f value is $\leq f$ value of current state, return the current state.
- Think of this as looking for a local max of f.
- The above assumes you're look for state with high f value. Change accordingly if you want a state with low f value.

n-Queens Problem I

- Problem: On 8x8 chess place 8 queens so that every pair is non-attacking.
  - You can generalize this to the n-queen problem.
  - We want a **state**, not a path.
- States: board with 1 queen in each column.
- Successor: move one queen to a different row within the same column.
  - $8 \times 7 = 56$ successor states for each state.
- h: heuristic cost is number of attacking pairs.
  - You want to minimal h, i.e., $h(node) = 0$.

n-Queen Problem

Problems with Hill Climbing

- You might go along a path where the largest f-value is not present (or smallest h value).
- Here's an 8-queens state where h=1 but every successor has $h > 1$ ... STUCK!!!

Variations of Hill Climbing I

- **Sideways**: allow choosing successor of same f-value. Need to take care not to go into infinite loop. For instance maximum number of + "sideways" moves.
- **Stochastic hill climbing**: randomly picks among successors with f-values greater than current (need not be the largest).
  - Configure the selection so that a state with greater change in f value means higher probability of being selected.

Variations of Hill Climbing I

- **First-choice hill climbing**: randomly generates one successor at a time and return when one with f-value better than current is found.
  - Example: For the case of n-queens problem, randomly select a column and then randomly select a new row for the queen of that column.
  - Use this if set of successors too large.

Variations of Hill Climbing I

- **Random-restart hill climbing**: randomly generate **initial state** and run hill climbing until goal is found.
  - Parameters include when to restart and how many restarts.
  - Example: Generate one random initial state, run hill climbing until stuck - get x. If x is desired result – DONE. Otherwise, restart with random initial state, run until stuck - get y. Keep better of x,y. Repeat.
  - Example: Same as above, but run for a fixed amount of time before restart.

Variations of Hill Climbing I

- Random-Restart Hill-Climbing Continued:
  - Example: Same as above but restart when f improves too slowly.
  - Number of restarts can be fixed or “continue as long as solution is not found”.
- Random restart is very simple but usually very successful (and fast).

Variations of Hill Climbing I

- **Simulated annealing search**:
  - (See textbook for connection with metallurgy and VLSI layout problem)
  - First pick a random successor.
  - If successor has better f-value, accept; otherwise accept random successor probabilistically:
    - Probability decreases exponentially depending on difference of f-values of current and successor.
    - Probability decreases as process continues.

Variations of Hill Climbing I

```
ALGORITHM: Simulated annealing
INPUT:
    schedule: function of time to "temp"
    f: max objective function
let x be a random state of the problem
t = 0 # basically time
while 1:
    T = schedule(t) # basically temperature
    if T is 0: return x
    y = random next state of x
    DE = f(y) - f(x)
    if DE > 0:
        x = y
    else:
        x = y with probability e^(DE/T)
    t += 1
```

Variations of Hill Climbing II

- T decreases as t increases
- Simulated annealing can be slow.

Online and Offline Search I

- All the informed and uninformed search algorithms do not probe the environment for changes.
- It receives an initial state, and then compute the path of actions.
- These algorithms are called **offline search algorithms**
- **Online search algorithms**: agents running the software/algorithm need to continually take an action, check the state of environment.
- In many problems, the agent does not even know the environment or search space at all. The only way to find all successors is by trying all actions.