Posts and Telecommunication Institute of Technology
Faculty of Information Technology 1

Introduction to Artificial Intelligence

Local search

Dao Thi Thuy Quynh

Outline

- Introduction to local search
- Hill-climbing search
- Simulated annealing algorithm

Local search

- Previous search algorithms (blind or informed) investigate the search space systematically according to certain rules
  - Need to store information about traversed states and paths
  - Not suitable for problems with large state spaces

- Local search only considers the current state and its neighbors
  - Does not store information about traversed states and paths
  - Saves time and memory
  - Applicable to problems with large state spaces
  - Does not give optimal solution

Combinatorial optimization problem

- Find an optimal state or an optimal combination in a discrete space of states
  - Do not care about the path
- Huge state space
  - Can't use learned search methods to traverse all states
- There is no algorithm that allows to find the optimal solution with small computational complexity
  - Accept a relatively good solution
- Examples: planning, scheduling, etc.

Local search: Idea

- Local search only cares about goal state (optimal state), doesn't care about paths
  - Each state is a solution (sub-optimal)

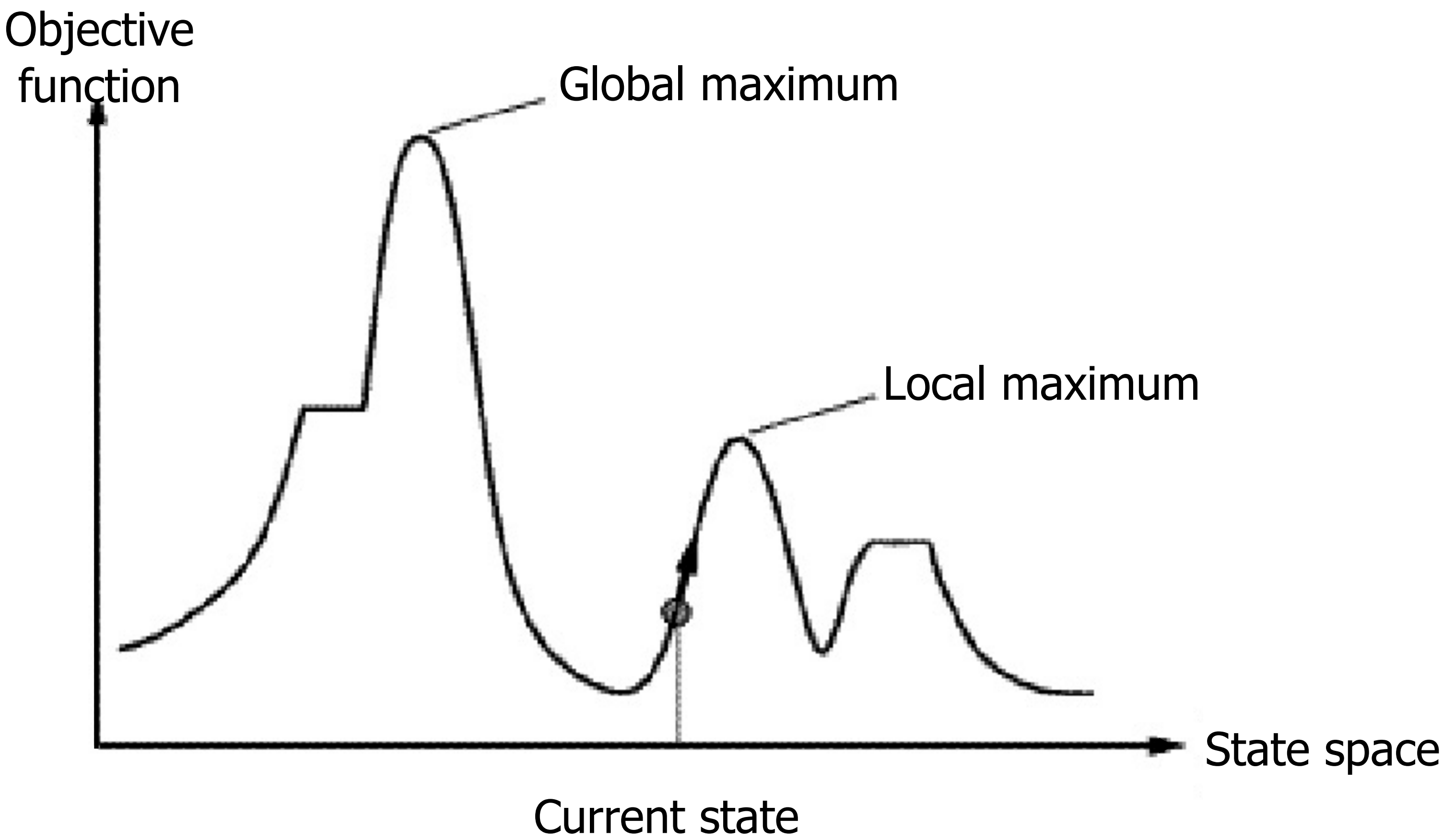
- Iteratively improve the solution by starting from one state, then moving to another state with a better objective function value

- Moving to other states by taking actions
  - A state obtained by taking an action from sate n is call a neighbor of n

Local search problem

- State space X
- Objective function $Obj: X \rightarrow R$
- Set of actions (to generate neighbors)
  - $N(x)$ set of neighbors of x
- Task: Find state x^* such that $Obj(x^*)$ is maximum (or minimum)

Local search



Outline

- Introduction to local search
- Hill-climbing search
- Simulated annealing algorithm

Hill-climbing algorithm: Idea

- **Hill-climbing**: Name of a family of algorithms with the same principle
- **Principle**: From the current state, consider the set of neighbors, move to a better state
  - How to choose a neighbor to move?
- **Goal state**: The algorithm stops when there is no better neighbor
  - Algorithm can find the global maximum or a local maximum

Moving to the best state

Input: combinatorial optimization problem

Output: state with the maximum value of the objective function (or local maximum)

1. Select a random state x
2. Let Y denotes the set of neighbors of x
3. **if** $\forall y_i \in Y: Obj(y_i) < Obj(x)$
   - **return** x
4. $x \leftarrow y_i$ where $i = argmax_i\,(Obj(y_i))$
5. **Go to** 2

Moving to the best state

Properties of hill-climbing algorithm

- Simple, easy to implement
- Less memory (does not store states)
- Can get stuck on local maxima (sub-optimal)
- Selecting the set of actions is important; there are no general rules
  - If the number of actions is large
    - Generate many neighbors
    - Take time to choose the best neighbor
  - If the number of actions is small
    - Usually give a local maximum

Random hill-climbing: idea

- Another version of the hill-climbing algorithm

- Randomly select a neighbor
  - Move to this neighbor if it is better (than the current state)
  - If it isn't better, randomly select another neighbor

- End when patience runs out
  - Number of selected neighbors (in a loop or in the algorithm)

Random hill-climbing

1. Select a random state x
2. Let Y denote the set of neighbors of x
3. Select a random state $y_i \in Y$
4. **if** $Obj(y_i) > Obj(x)$
   - $x \leftarrow y_i$
5. **Go to** 2 if still have patience

Problem: How to choose the ending condition?

Some properties

- When each state has many neighbors
  - Random hill-climbing usually gives results faster and has more chance to find the global maximum (than "moving to the best state" version)
- For state spaces with few local maxima
  - Hill-climbing algorithms usually find a solution quickly
- For complex state spaces
  - Hill-climbing algorithms usually find a local maximum
  - Random-restart hill-climbing can find a good local maximum

Outline

- Introduction to local search
- Hill-climbing search
- Simulated annealing algorithm

Simulated annealing: Idea

- A generalized version of random hill-climbing

- **Goal**: partially solve the problem of local maxima in hill-climbing algorithms

- **Principle**: accept states that are worse than the current state with a probability p
  - How to choose the value of p?

Choosing p

- **Principle**: does not choose a fix value for p; the value of p depends on two factors
  - If the new state is much worse than the current state, p must be decreased
    - The probability of accepting a state is inversely proportional to the poorness of the state
  - Over time, the value of p must be decreased
    - At the beginning, the algorithm is not in a good state region, a big change can be accepted
    - Over time, the algorithm is in a good state region, the change should be limited

Simulated annealing algorithm

$SA(X, Obj, N, m, x, C)$

Input: number of iterations m

initial state x (random)

cooling schedule C

Output: optimal state x^*(maximize objective function)

Initialize:$x^* = x$

for $i = 1$ **to** m

1. randomly select $y \in N(x)$

   a) calculate $\Delta(x, y) = Obj(x) - Obj(y)$

   b) **if** $\Delta(x, y) < 0$ **then** $p = 1$

   c) **else** $p = e^{-\Delta(x,y)/T}$

   d) **if** $rand[0,1] < p$ **then** $x \leftarrow y$

   if $Obj(x) > Obj(x^*)$ **then** $x^* \leftarrow x$

2. decrease T by using schedule C

return x^*

Cooling schedule C

- $T_t = T_0 * \alpha^{t*k}$
  - $T_0 > 0$
  - $\alpha \in (0,1)$
  - $1 \le t \le m$
  - $1 < k < m$

- Meaning
  - Increase t; small T; small p
  - Large T: accept every states
    - Random walk
  - Small T: does not accept worse states
    - Random hill-climbing

Properties of simulated annealing

- No clear theoretical foundations

- Usually gives better results than hill-climbing algorithms
  - Reduce the possibility of giving local maxima

- Parameter selection depends on specific problems