Artificial Intelligence & Machine Learning

Bodhayan Roy

Department of Mathematics,
Indian Institute of Technology Kharagpur

Lecture 6

Local Search

- Mostly we have to search in the solution space instead of state space, and optimize an objective function.
- Proceed through nearby solutions.

Hill-climbing search

function HILL-CLIMBING(problem) returns a state that is a local maximum

current $\leftarrow$ MAKE-NODE(problem.INITIAL-STATE)

loop do

neighbor $\leftarrow$ a highest-valued successor of current

if neighbor.VALUE $\leq$ current.VALUE then return current.STATE

current $\leftarrow$ neighbor

Hill-climbing search

- Stochastic hill climbing chooses at random from among the uphill moves.
- First-choice hill climbing implements stochastic hill climbing by generating successors randomly until one is generated that is better than the current state.
- Random-restart hill climbing adopts the well-known saying, “If at first you don’t succeed, try, try again.”

Simulated annealing

function SIMULATED -ANNEALING(problem, schedule) returns a solution state
  inputs: problem,
    schedule, a mapping from time to "temperature"
  current $\leftarrow$ MAKE -NODE(problem.INITIAL-STATE)
  for t $=$ 1 to ∞ do
    T $\leftarrow$ schedule(t)
    if T $=$ 0 then return current
    next $\leftarrow$ a randomly selected successor of current
    ΔE $\leftarrow$ next.VALUE – current.VALUE
    if $\Delta E > 0$ then current $\leftarrow$ next
    else current $\leftarrow$ next only with probability $e^{\Delta E/T}$

Simulated annealing

The innermost loop of the simulated-annealing algorithm is quite similar to hill climbing. Instead of picking the best move, however, it picks a random move. If the move improves the situation, it is always accepted. Otherwise, the algorithm accepts the move with some probability less than 1. The probability decreases exponentially with the "badness" of the move—the amount ΔE by which the evaluation is worsened. The probability also decreases as the "temperature" T goes down: "bad" moves are more likely to be allowed at the start when T is high, and they become more unlikely as T decreases. If the schedule lowers T slowly enough, the algorithm will find a global optimum with probability approaching 1.

Genetic algorithm

function GENETIC -ALGORITHM(population , FITNESS -FN)
returns an individual
  inputs: population , a set of individuals
    FITNESS -FN, a function that measures the fitness of an
individual repeat
   new population ← empty set
   for i = 1 to SIZE(population) do
    x ← RANDOM -SELECTION (population, FITNESS -FN)
    y ← RANDOM -SELECTION (population , FITNESS -FN)
    child ← REPRODUCE (x , y)
    if (small random probability) then child ← MUTATE(child)
    add child to new population
population ← new population
until some individual is fit enough, or enough time has elapsed
return the best individual in population , according to FITNESS
-FN

Genetic algorithm

function REPRODUCE (x,y) returns an individual
    inputs: x, y, parent individuals

    n $\leftarrow$ LENGTH(x); c $\leftarrow$ random number from 1 to n
    return APPEND(SUBSTRING(x,1,c), SUBSTRING(y, c + 1, n))