Artificial Intelligence

BS (CS) _SP_2025

Lab_06 Manual

Learning Objectives:

1. Simulated Annealing
2. Local Beam Search

1. Simulated Annealing:

Simulated annealing is a stochastic optimization algorithm that is often used to find the global minimum (or maximum) of a non-convex function. The algorithm is inspired by the process of annealing in metallurgy, where a metal is heated and then slowly cooled to reduce its defects and improve its properties.

Simulated Annealing explores the solution space by allowing occasional uphill moves (worse solutions) to escape local optima. This is controlled by a temperature parameter that decreases over time, reducing the likelihood of accepting worse solutions as the search progresses.

Step-by-Step Working:

1. **Initialize:** Start with an initial solution and a high temperature.
2. **Evaluate:** Compute the cost of current solution.
3. **Generate neighbor:** Select a neighboring solution randomly.
4. **Compute cost difference:** measure the difference between new solution and current solution.
5. **Acceptance Criteria:**
    - if new solution is better, accept it.
    - if new solution is worse, accept it with a probability dependent on temperature.
6. **Cool Down:** reduce the temperature gradually.
7. **Repeat:** continue until the system is sufficiently cooled down or a stopping condition is met.

Components of Simulated Annealing:

- Initial Solution → a random point in search space.
- Neighbor Selection → a method to select next solution/node.
- Cost Function → method to evaluate the quality of solution.
- Acceptance probability → decides whether to move to a worse solution or not using Boltzmann probability: $P = e^{-\frac{\Delta E}{T}}$
- Cooling Schedule → determines how temperature decreases over time. Usually changed slowly by a fixed fraction.

```
Simulated_Annealing (Problem, Initial_Temperature, Cooling_Rate)
  current_solution  = Initial_Solution(Problem) --- randomly generated initial solution
  current_temperature = Initial_Temperature
  while stopping condition not met do
     new_solution = Generate_Neighbor(current_solution)
     cost_difference = Cost(new_solution) - Cost(current_solution)
     if cost_difference < 0 then
        current_solution = new_solution
     else
        probability = exp (-cost_difference / current_temperature)
        if Random (0,1) < probability then
           current_solution = new_solution
     current_temperature = current_temperature * Cooling_Rate
  return current_solution
```

2. Local Beam Search:

Local Beam Search is a heuristic search algorithm that explores multiple candidate solutions simultaneously. Unlike Simulated Annealing or Hill Climbing, which follow a single solution path, Local Beam Search maintains a fixed number of "k" best solutions and expands them in parallel. This approach helps in escaping local optima by leveraging information from multiple promising solutions.

Step-by-Step Working:

1. **Initialize:** Start with "k" randomly generated solutions.
2. **Evaluate:** Compute the cost (or heuristic value) for each solution.
3. **Generate Successors:** Expand each solution to produce new neighboring states.
4. **Select Best k Candidates**: From all generated successors, keep only the "k" best solutions.
5. **Repeat**: Continue until a stopping condition is met (e.g., a solution is good enough or a maximum number of iterations is reached).

```
Local_Beam_Search(Problem, k)
  Initialize k random solutions
  while stopping condition not met do
     Generate all possible successors of the k solutions
     Select the best k successors based on evaluation function
  return best found solution
```