Chapter 1

Local Search Algorithms

1.1 "Hill Climbing" (Greedy Local)

Assumption: We don't go to a repeated state

Do we always need to care about the path to the goal?

1.2 Informed Search: the Idea

When traversing the search tree, use domain knowledge / heuristics to avoid search paths (moves/actions) that are likely to be fruitless

1.3 Informed Search and Heuristics

Informed search relies on domain-specific knowledge/hints that help locate the goal state.

1.4 Hard Problems

- Many important problems are provably not solvable in polynomial time (NP and harder)
- Results based on the worst-case analysis
- In practice: instances are often easier
- Approximate methods can often obtain good solutions

1.5 Local Search

- Moves between configurations by performing local moves
- Works with complete assignments of the variables

- Optimization problems:
  - Start from a suboptimal configuration
  - Move towards better solutions
- Satisfaction problems:
  - Start from an infeasible configuration
  - Move towards feasibility
- NO guarantees
- Can work great in practice!

1.6 Local Search Algorithms

If the path to the goal does not matter, we might consider a different class of algorithms. Local Search Algorithms

- do not worry about paths at all.
- Local Search

1.6.1 Selecting Neighbor

- How to select the neighbor?
  - exploring the whole or part of the neighborhood
- Best neighbor
  - Select "the" best neighbor in the neighborhood
- First neighbor
  - Select the first "legal" neighbor
  - Avoid scanning the entire neighborhood
- Multi-stage selection
  - Select one "part" of neighborhood and then
  - select from the remaining "part" of the neighborhood
- Hill-climbing search
  - Gradient descent in continuous state spaces
  - Can use e.g. Newton's method to find roots

- Simulated annealing search
- Tabu search
- Local beam search
- Evolutionary/genetic algorithms

Although local search algorithms are not systematic, they have two key advantages:

- they use very little memory – usually a constant amount; and
- the can often find reasonable

Local search algorithms are useful for search pure optimization problems, in which the aim is to find the best state according to the objective function

1.7 Simulated Annealing

1.7.1 What Is It?

In metallurgy, annealing is the process used to temper or harden metals and glass by heating them to a high temperature T and then gradually cooling them, thus allowing the material to coalesce into a low-energy E crysalline state (less or no defects).
Key Idea:

- Use Metropolis algorithm but adjust the temperature dynamically
- Start with a high temperature (random moves)
- Decrease the temperature
- When the temperature is low, becomes a local search

1.8 Metropolis Heuristics

1.8.1 Basic Idea

- Accept a move if it improves the objective value
- Accept "bad moves" as well with some probability
- The probability depends on how "bad" the move is
- Inspired by statistical physics

1.8.2 How to choose the probability?

- t is a scaling parameter (called temperature)
- Δ is the difference $f(n) - f(s)$

1.8.3 Fixed T

- What happens for a large T?
  - Probability of accepting a degrading move is large
- What happends for a small T?
  - Probability of accepting a degrading move is small

Algorithm 1.1 Simulated Annealing Pseudocode

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
    $current \leftarrow problem$.INITIAL
    $t \leftarrow 1$
    while True **do**
        $T \leftarrow$ SCHEDULE(t)
        if $T == 0$ **then return** *current*
        end if
        $next \leftarrow$ a randomly selected successor of *current*
        $\Delta E \leftarrow$ VALUE(*current*) − VALUE(*next*)
        if $\Delta E > 0$ **then**
            $current \leftarrow next$
        else
            $current \leftarrow next$ only with probability $e^{-\Delta E/T}$
        end if
    end while
end function

1.8.4 Temperature/Cooling Schedule

Idea: start with

1.8.5 Summary

- Converges to a global optimum
  - connected neighborhood
  - slow cooling schedule
    - *slower than the exhaustive search*

- In practice
  - can give excellent results
  - need to tune a temperature schedule
  - default choice: $t_{k+1} = \alpha t_k$
- Additional tools
  - restarts and reheats

1.8.6 Applications

- Basic Problems
  - Traveling salesman
  - Graph partitioning
  - Matching problems
  - Graph coloring
  - Scheduling
- Engineering
  -

1.9 Heuristics and Metaheuristics

- Heuristics
  - how to choose the next neighbor?
  - use local information (state and its neighborhood)
  - direct the search towards a local min/maximum
- Metaheuristics
  - how to escape local minima?
  - direct the search towards a global min/maximum
  - typically include some memory or learning