Huristic Algorithms: From Hill Climbing to Simulated Annealing: A Probabilistic Approach to Global Optimization

Advanced Algorithms
Department of Computer Science and Engineering, RGIPT

Contents

1 Introduction

Optimization problems arise in almost every branch of Computer Science and Engineering — from finding the shortest path, minimizing cost, maximizing throughput, or optimizing design parameters.

However, real-world objective functions are often **non-linear, discontinuous, and high-dimensional**. Traditional methods such as calculus-based optimization may fail or become computationally infeasible.

This lecture explores two important heuristic techniques:

1. **Hill Climbing (HC)** – a simple local search strategy.
2. **Simulated Annealing (SA)** – a probabilistic extension that can escape local minima.

Before understanding them, we first need to discuss the concept of convexity.

2 Convex and Concave Problems

2.1 Convex Function

A function $f(x)$ is said to be **convex** if for any two points x_1, x_2 in its domain and any $\lambda \in [0, 1]$:

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

This means that the line segment between any two points on the curve lies **above or on the curve**.

Implication: Convex functions have a single global minimum. Any local minimum is also the global minimum.

Example:

$$f(x) = x^2$$

is convex because its curvature is upward everywhere.

2.2 Concave Function

Similarly, $f(x)$ is **concave** if:

$$f(\lambda x_1 + (1 - \lambda)x_2) \geq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

This means the curve bends downward.

Example:

$$f(x) = -x^2$$

is concave and has a single global maximum.

2.3 Non-Convex Problems

In real-world scenarios, most problems are **non-convex**:

$$f(x) = x^2 + 4\sin(5x)$$

Such functions have multiple local minima and maxima — resembling a mountain range.

Finding the **global optimum** in such landscapes is challenging and forms the basis of heuristic methods like Hill Climbing and Simulated Annealing.

3 Hill Climbing Algorithm

3.1 Basic Idea

Hill Climbing is an **iterative improvement algorithm** that starts with a random solution and repeatedly moves to a better neighboring solution.

It can be thought of as:

> "A hiker climbing uphill (for maximization) or going downhill (for minimization) who only moves if the next step improves the view."

3.2 Pseudocode

```
Hill_Climbing(f, x0):
    current = x0
    while True:
        neighbor = best_neighbor(current)
        if f(neighbor) <= f(current):
            return current  # Local optimum found
        current = neighbor
```

3.3 Example

Minimize:

$$f(x) = x^2 + 4\sin(5x), \quad x \in [-10, 10]$$

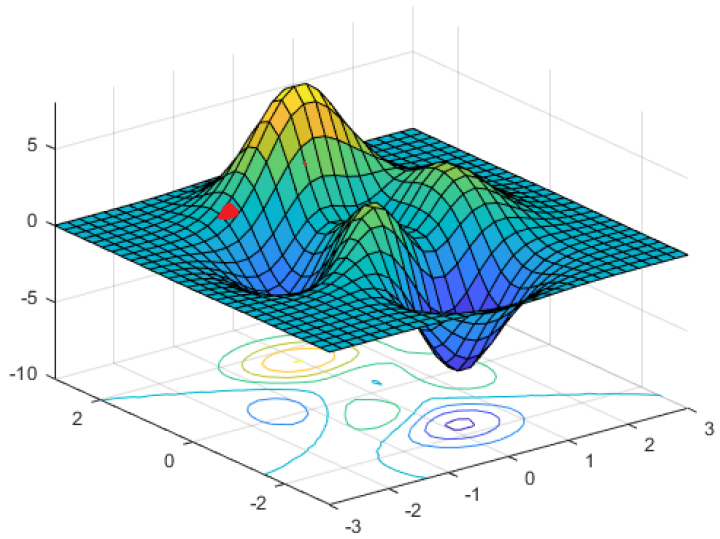
At each iteration:

- Generate two neighbors $x + \delta$ and $x - \delta$.
- Move to the neighbor with lower $f(x)$.

Observation: The algorithm often stops early at a **local minimum**, not the true (global) minimum.

4 Local Maxima and Minima Problem

Consider a non-convex landscape:



(A conceptual plot showing local minima and the global minimum.)

A point x_L is a **local minimum** if:

$$f(x_L) \leq f(x) \quad \forall x \text{ in the neighborhood of } x_L$$

But $\exists$ some x_G such that $f(x_G) < f(x_L)$.

Hence: Hill Climbing may get **stuck** in x_L, unable to explore further.

5 Simulated Annealing: Motivation and Analogy

In physics, **annealing** is a process where a material is heated to a high temperature and then slowly cooled. This allows atoms to move freely initially (exploration) and gradually settle into a stable, low-energy configuration (exploitation).

Optimization Analogy:

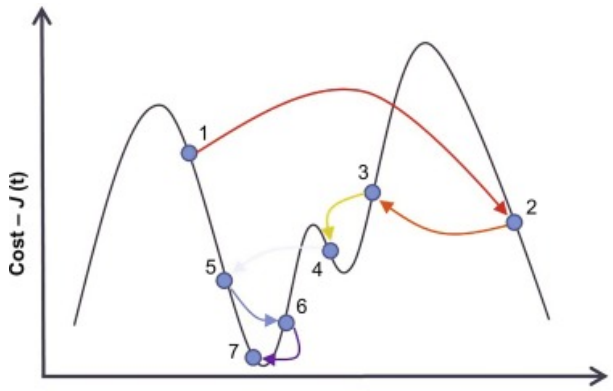
- **Temperature (T)** $\rightarrow$ controls randomness.
- **Energy (E)** $\rightarrow$ cost function $f(x)$.

Initially, high temperature allows occasional acceptance of worse solutions to explore widely. As the system cools, it becomes more selective, converging to a near-optimal solution.

6 Simulated Annealing Algorithm

6.1 Algorithm Steps

1. Initialize a random solution x and set initial temperature T_0.
2. Repeat until T reaches a minimum threshold:
   2.1. Select a random neighbor x'.
   2.2. Compute change in cost:
   $$\Delta E = f(x') - f(x)$$
   2.3. If $\Delta E < 0$, accept x' (better solution).
   2.4. If $\Delta E > 0$, accept x' with probability:
   $$P = e^{-\Delta E/T}$$
   2.5. Update temperature:
   $$T = \alpha T, \quad \text{where } 0 < \alpha < 1$$
3. Return the best solution found.



(An example of Simulated Annealng and how the jumps slows down gradually.)

6.2 Acceptance Probability in Simulated Annealing

Simulated Annealing (SA) uses the **Metropolis acceptance criterion** to decide whether to accept a new solution that is worse than the current one. This is the key mechanism that allows the algorithm to escape from local minima.

6.3 Metropolis Criterion

When moving from a current state S (energy E_{current}) to a new state S' (energy E_{new}), the change in energy (or cost) is given by:

$$\Delta E = E_{\text{new}} - E_{\text{current}}$$

The acceptance probability $P(\text{accept})$ is defined as:

$$P(\text{accept}) = \begin{cases} 1, & \text{if } \Delta E \leq 0 \quad \text{(better move)} \\ e^{-\Delta E/T}, & \text{if } \Delta E > 0 \quad \text{(worse move)} \end{cases}$$

where:

- T is the current **temperature parameter**, which gradually decreases with time.
- e is the base of the natural logarithm ($e \approx 2.718$).

At high temperatures (T large), the algorithm frequently accepts worse moves, enabling exploration. At low temperatures, it becomes more selective and primarily accepts better moves, promoting convergence.

6.4 Numerical Examples

Example 1: High Temperature

$$E_{\text{current}} = 100, \quad E_{\text{new}} = 105, \quad T = 50$$

$$\Delta E = 5 \quad \Rightarrow \quad P(\text{accept}) = e^{-5/50} = e^{-0.1} \approx 0.9048$$

Thus, there is approximately a **90% chance** of accepting this worse move.

Example 2: Medium Temperature

$$E_{\text{current}} = 100, \quad E_{\text{new}} = 105, \quad T = 10$$
$$\Delta E = 5 \quad \Rightarrow \quad P(\text{accept}) = e^{-5/10} = e^{-0.5} \approx 0.6065$$

Here, there is about a **60% chance** of accepting the worse move.

Example 3: Low Temperature

$$E_{\text{current}} = 100, \quad E_{\text{new}} = 105, \quad T = 1$$
$$\Delta E = 5 \quad \Rightarrow \quad P(\text{accept}) = e^{-5/1} = e^{-5} \approx 0.0067$$

Now, the probability of acceptance is very low (**0.67%**), so the algorithm becomes almost greedy.

6.5 Effect of ΔE and T

ΔE	Temperature T	$P(\text{accept}) = e^{-\Delta E/T}$	Interpretation
5	50	0.9048	Hot system: easily accepts bad move
5	10	0.6065	Moderately exploratory
5	1	0.0067	Nearly frozen, behaves greedily
20	10	0.1353	Large cost increase, rarely accepted

6.6 Decision Rule Implementation

In practice, the acceptance decision is made stochastically:

1. Compute $\Delta E = E_{\text{new}} - E_{\text{current}}$.
2. If $\Delta E < 0$, accept the move (improvement).
3. Otherwise, generate a random number $r \in [0, 1]$.
4. Accept the move if $r < e^{-\Delta E/T}$, else reject.

Example: Simulated Steps

Step	E_{current}	E_{new}	T	ΔE	$P(\text{accept})$	Decision
1	100	105	50	5	0.9048	Accepted
2	105	107	10	2	0.8187	Rejected (if $r = 0.9$)
3	105	103	10	-2	—	Accepted (better move)

6.7 Interpretation and Intuition

- At **high temperature**, the system behaves like a random explorer, willing to jump over local minima.
- As **temperature decreases**, the system becomes more selective, focusing on refining the current solution.
- This transition from exploration to exploitation enables SA to escape local minima and converge toward a global minimum.

6.8 Analogy

The process resembles the behavior of atoms in a heated metal:

- At high temperature, atoms vibrate freely and can rearrange into many configurations (exploration).
- As the metal cools, atomic motion slows, and atoms settle into a low-energy, stable configuration (optimal solution).

6.9 Key Takeaways

- ΔE: Determines how much worse the new solution is.
- T: Controls the willingness to accept worse solutions.
- $e^{-\Delta E/T}$: Governs acceptance probability.
- High T: High randomness (exploration).
- Low T: Low randomness (exploitation).
- Random acceptance: Prevents getting trapped in local minima.

7 Example: Traveling Salesman Problem (TSP)

Objective: Find the shortest possible route visiting n cities exactly once and returning to the starting point.

7.1 Representation

A route is a permutation of city indices, e.g.:

$$(1, 3, 2, 5, 4, 1)$$

7.2 Cost Function

$$E(S) = \sum_{i=1}^{n-1} d(C_i, C_{i+1}) + d(C_n, C_1)$$

7.3 Neighbor Generation

Swap two cities randomly to create a new route.

7.4 Pseudo-code (TSP version)

```
route = random_permutation(cities)
best = route
T = 10000
alpha = 0.995
while T > 1e-3:
    new_route = swap_two_cities(route)
    E = cost(new_route) - cost(route)
    if E < 0 or random() < exp(-E/T):
        route = new_route
    if cost(route) < cost(best):
        best = route
    T = alpha * T
return best
```

This process effectively explores different permutations and converges to a near-optimal tour.

8 Cooling Schedules

Temperature is reduced according to a schedule:

Type	Formula	Comment
Exponential	$T = T_0 \times \alpha^k$	Simple and effective
Linear	$T = T_0 - k\beta$	May cool too quickly
Logarithmic	$T = \frac{T_0}{\log(1+k)}$	Very slow but theoretically optimal

Typical values: $\alpha = 0.95$ to 0.999.

9 Comparison: Hill Climbing vs Simulated Annealing

Aspect	Hill Climbing	Simulated Annealing
Determinism	Deterministic	Probabilistic
Escape from local minima	No	Yes, via probability
Exploration vs Exploitation	Only exploitation	Both, controlled by T
Parameter	Step size	Temperature, cooling rate
Best for	Smooth functions	Complex, multimodal landscapes

10 Key Insights and Discussion

10.1 Advantages of SA

- Can escape local minima.
- Works for both discrete and continuous problems.
- Simple and flexible to implement.

10.2 Limitations

- Requires tuning of parameters (T_0, α, iterations).
- Slower than greedy methods.
- No guarantee of exact global optimum.

11 Extensions and Research Directions

1. **Adaptive Cooling:** Adjust α dynamically based on improvement rate.
2. **Hybrid Algorithms:** Combine SA with Genetic Algorithms or Tabu Search.
3. **Parallel Simulated Annealing:** Use multiple temperatures on distributed processors.
4. **Quantum Annealing:** Physical realization of annealing on quantum computers (e.g., D-Wave systems).
5. **Machine Learning Optimization:** Use SA for feature selection or neural network weight tuning.

Each of these directions aims to improve convergence speed or global accuracy by balancing exploration and exploitation more efficiently.

12 Summary and Takeaways

- Convex problems have a single global optimum; non-convex ones do not.
- Hill Climbing is simple but gets stuck in local minima.
- Simulated Annealing introduces controlled randomness using a temperature parameter.
- Acceptance probability follows the Boltzmann distribution:

$$P = e^{-\Delta E/T}$$

- The algorithm transitions from exploration to exploitation as temperature cools.
- SA has widespread applications: scheduling, routing, engineering design, ML, and more.

In summary: Simulated Annealing bridges physics and computation — transforming the process of metal cooling into a powerful method for solving some of the most challenging optimization problems in computer science.