Local Search and Hill Climbing

- We have focused so far on **stochastic hill climbing**.
- A pure hill climber can easily get stuck in **local optima**.
- Restarting the search can help escape local optima, but restarting is **costly** since it begins from scratch.
- We can remember the best-ever solution, but the search still wastes time rediscovering good areas.
- Using larger neighborhoods (e.g., `nswap`) can help, but:
  - too small $\rightarrow$ still stuck in local optima
  - too large $\rightarrow$ search becomes slow or inefficient
- Therefore, we often use a simple operator like `1swap`.

Idea Behind Improvement

- A local optimum under `1swap` is usually **similar** to the global optimum.
- However, it also differs slightly — otherwise it would already be the global optimum.
- The difference is such that the `1swap` operator cannot fix it.
- When we restart, we **lose the similarities** to the global optimum that we already discovered.
- Then, the algorithm must rediscover these similarities again.
- This makes the process inefficient — can we find a **less costly way** to improve from the current solution?

Algorithm Concept: Probabilistic Acceptance of Worse Solutions

Metallurgy: What is Annealing?

- In metallurgy, **annealing** is a heat treatment to reach a low-energy, stable crystal structure.
- Process:
  1. Heat the metal: atoms move freely; defects can rearrange.
  2. **Slowly cool**: atoms settle into a **lower-energy** configuration.
- If cooling is too fast, defects get “frozen in” (a suboptimal structure).

Mapping Metallurgy to Optimization

Metallurgy	**Simulated Annealing (SA)**
Atoms	Candidate solutions
Energy of the crystal	Objective / cost function
Temperature (heat)	Randomness / acceptance of worse moves
Slow cooling	Temperature schedule $T(t)$
Stable low-energy structure	Near-global optimum

Key Idea of Simulated Annealing

- Start "hot": allow big, random moves $\Rightarrow$ broad exploration.
- Gradually **cool**: reduce randomness, favor better moves.
- Unlike pure descent, SA can **escape local minima** by sometimes accepting worse solutions.

Simulated Annealing (SA)

- **Simulated Annealing (SA)** is a local search method designed to **escape local optima**.
- Instead of restarting when stuck, it sometimes accepts **worse solutions** to explore new areas.
- The idea is inspired by the **annealing process in metallurgy**, where materials are slowly cooled to reach a stable structure.

Main Principles:

1. Worse solutions can be accepted temporarily.
2. Acceptance probability P decreases as the quality difference ΔE increases.
3. P also decreases over time (fewer bad moves as the search continues).

- These rules are added into the **hill climbing loop**.
- **Question:** How can we implement these concepts?

Acceptance Probability in Simulated Annealing

- Now, let's see how to **implement** the acceptance rule in SA.
- Assume the current solution is x and we generate a new solution x' using our local search operator.
- The change in the objective value is denoted as:

$$\Delta E = f(\gamma(x')) - f(\gamma(x))$$

- $\Delta E < 0 \Rightarrow x'$ is **better** than x.
- $\Delta E > 0 \Rightarrow x'$ is **worse**.
- $\Delta E = 0 \Rightarrow$ both have the **same quality**.

Intuition

SA allows occasional acceptance of worse solutions ($\Delta E > 0$) to escape local optima, depending on the probability function and temperature.

Acceptance Probability in SA

- The change in the objective value is:

$$\Delta E = f(\gamma(x')) - f(\gamma(x))$$

- The probability P to accept a new (possibly worse) solution x' is:

$$P = \begin{cases} 1, & \text{if } \Delta E \leq 0 \\ e^{-\frac{\Delta E}{T}}, & \text{if } \Delta E > 0 \text{ and } T > 0 \\ 0, & \text{otherwise (when } \Delta E > 0 \text{ and } T = 0) \end{cases}$$

Interpretation:

- If the new solution is better ($\Delta E \leq 0$), we always accept it.
- If the new solution is worse ($\Delta E > 0$):
  - The acceptance probability **decreases** as ΔE increases.
  - It also **decreases** as the temperature T gets smaller.

Acceptance Probability in SA

- The change in the objective value is:

$$\Delta E = f(\gamma(x')) - f(\gamma(x))$$

- The probability P to accept a new (possibly worse) solution x' is:

Interpretation:

- If the new solution is better ($\Delta E \leq 0$), we always accept it.
- If the new solution is worse ($\Delta E > 0$):
  - The acceptance probability **decreases** as ΔE increases.
  - It also **decreases** as the temperature T gets smaller.

Key Idea

A higher temperature T makes the algorithm more **explorative**, while a lower T makes it more **selective**.

Temperature Schedule

Temperature Schedule in Simulated Annealing

- The acceptance probability depends on the **temperature** $T(\tau)$:

$$P = \begin{cases} 1, & \text{if } \Delta E \leq 0 \\ e^{-\frac{\Delta E}{T(\tau)}}, & \text{if } \Delta E > 0 \text{ and } T(\tau) > 0 \\ 0, & \text{otherwise} \end{cases}$$

How temperature works:

- $T(\tau)$ decreases gradually as iterations τ increase.
- Initially, the system is "**hot**" $\Rightarrow$ even significantly worse solutions may be accepted.
- As the process "**cools down**," $T(\tau)$ gets smaller, and only slightly worse or better solutions are accepted.
- When $T(\tau) = 0$, the algorithm behaves like pure hill climbing — only better solutions are accepted.

Key Idea

$T(\tau)$ is a **monotonically decreasing function** which defines the **temperature schedule**.

Conditions for Temperature Schedule

$$\text{The acceptance probability } P = \begin{cases} 1, & \text{if } \Delta E \leq 0 \\ e^{-\frac{\Delta E}{T(\tau)}}, & \text{if } \Delta E > 0 \text{ and } T(\tau) > 0 \\ 0, & \text{otherwise} \end{cases}$$

Conditions for the Temperature Function $T(\tau)$:

- $T(\tau)$ **decreases gradually** as the number of iterations τ increases.
- It approaches zero: $\lim_{\tau \to \infty} T(\tau) = 0$.
- It starts with an initial temperature T_s when $\tau = 1$.
- Apart from these conditions, we can design $T(\tau)$ in various ways (e.g., linear, exponential, or logarithmic cooling).

Key Point

The temperature schedule controls how quickly the search moves from **exploration** (high T) to **exploitation** (low T).

The Meaning of the Temperature Schedule

- Why do we use a **temperature schedule** in Simulated Annealing?
- It provides a way to balance two opposite goals:
  - **Exploration:** trying new and diverse solutions.
  - **Exploitation:** improving the current good solutions.
- At the beginning, the temperature T is **high**:
  - Many worse solutions are accepted.
  - The algorithm explores the search space widely (similar to random sampling).
- As time passes, T **decreases**:
  - Fewer bad moves are accepted.
  - The search focuses more on improving good solutions.
- At the end, $T \approx 0$:
  - Only better solutions are accepted — the algorithm behaves like hill climbing.

Key Insight

The temperature schedule controls the **transition from exploration to exploitation**, helping the algorithm find a good balance between discovering and refining solutions.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.
3. Generate a neighbor x' by slightly modifying x.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.
3. Generate a neighbor x' by slightly modifying x.
4. Increase $\tau = \tau + 1$.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.
3. Generate a neighbor x' by slightly modifying x.
4. Increase $\tau = \tau + 1$.
5. **If** x' is better than x_b, update $x_b = x'$.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.
3. Generate a neighbor x' by slightly modifying x.
4. Increase $\tau = \tau + 1$.
5. **If** x' is better than x_b, update $x_b = x'$.
6. **If** x' is better than x, update $x = x'$.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.
3. Generate a neighbor x' by slightly modifying x.
4. Increase $\tau = \tau + 1$.
5. **If** x' is better than x_b, update $x_b = x'$.
6. **If** x' is better than x, update $x = x'$.
7. **If** x' is worse ($\Delta E > 0$):
    - Accept it as the new x with probability $P(\Delta E, \tau) = e^{-\Delta E / T(\tau)}$,
    - Otherwise, keep the current x.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.
3. Generate a neighbor x' by slightly modifying x.
4. Increase $\tau = \tau + 1$.
5. **If** x' is better than x_b, update $x_b = x'$.
6. **If** x' is better than x, update $x = x'$.
7. **If** x' is worse ($\Delta E > 0$):
    - Accept it as the new x with probability $P(\Delta E, \tau) = e^{-\Delta E / T(\tau)}$,
    - Otherwise, keep the current x.
8. Repeat steps 3–7 until the stopping condition (time or iteration limit) is met.

Simulated Annealing Algorithm

- Simulated Annealing can be seen as: Hill Climbing + Probabilistic acceptance of worse solutions.
- Basic idea: Sometimes accept worse solutions to escape local optima.

1. Initialize iteration counter $\tau = 1$.
2. Generate a random initial solution x. Set both the current point x and the best point x_b to this value.
3. Generate a neighbor x' by slightly modifying x.
4. Increase $\tau = \tau + 1$.
5. **If** x' is better than x_b, update $x_b = x'$.
6. **If** x' is better than x, update $x = x'$.
7. **If** x' is worse ($\Delta E > 0$):
    - Accept it as the new x with probability $P(\Delta E, \tau) = e^{-\Delta E / T(\tau)}$,
    - Otherwise, keep the current x.
8. Repeat steps 3–7 until the stopping condition (time or iteration limit) is met.
9. Return the best solution x_b.