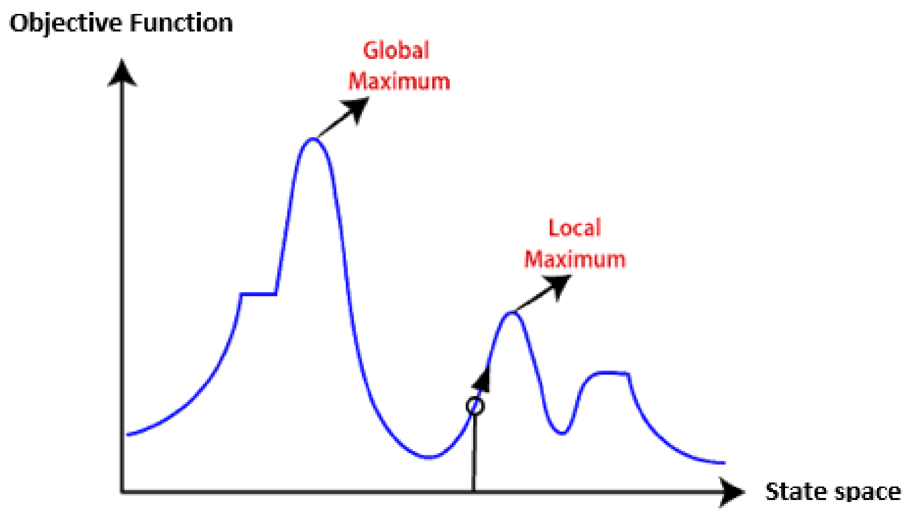
Local Search (Optimization)

Md Rifat Ahmmad Rashid, Associate Professor, EWU

*"In many hard problems the cost of finding **any** solution is modest, but the cost of finding the **best** one is astronomical. Local-search converts astronomical searches into a hike across the landscape of solutions."*



A one-dimensional state-space landscape in which elevation corresponds to the objective function

---

1 Why Local Search?

Table 1 contrasts global and local search approaches.

Table 1: Global versus local search in AI optimisation

Characteristic	**Global search (e.g. BFS, A*)**	**Local search**
Memory	Exponential in depth/breadth	Often $O(1)$–$O(n)$
Objective	*Find* a goal state	*Optimise* an objective function
Applicability	Discrete & continuous, but limited by branching	Very broad: scheduling, layout, neural-net tuning, robotics, etc.
Typical use-case	Path-finding, theorem proving	Hard combinatorial or numeric optimisation where admissible heuristics are unavailable

Local search assumes a *state* $\mapsto$ *cost/value* mapping. The agent iteratively *moves* to a *neighbour* state until a stopping criterion is met.

2 Canonical Algorithms

2.1 Hill-Climbing Family

See Table 2 for a comparison of variants.

Table 2: Variants of hill-climbing local search

Variant	**Key Idea**	**Pros**	**Cons**
Simple/Steepest-Ascent	Move to neighbour with greatest improvement	Easy; no parameters	Stuck at local maxima, plateaus, ridges
First-Choice	Examine random neighbours until an improvement is found	Good when neighbourhood is large	Same local-optimum issues
Stochastic	Pick a neighbour at random *weighted by* improvement	Adds exploration	Parameter tuning required
Random-Restart	Reapply hill-climb from random states	Probabilistically complete	Wastes time if optimum basin is small

Pseudocode (Steepest-Ascent).

```
function HILL_CLIMB(s):
    loop:
        best ← argmin{f(s’) | s’ in N(s)}
        if f(best)  f(s): return s   // no improvement
        s ← best
```

2.2 Simulated Annealing (SA)

Borrowed from statistical thermodynamics. A *temperature* T controls the probability of *up–hill* moves:

$$P(\text{accept } \Delta E) = \begin{cases} 1 & \Delta E \leq 0, \\ \exp\left(-\frac{\Delta E}{T}\right) & \Delta E > 0. \end{cases}$$

Typical cooling schedules are $T_{k+1} = \alpha T_k$ (*geometric*, $\alpha \approx 0.9$) or $T_k = \dfrac{T_0}{1 + \beta k}$ (*linear*). **Theorem (Geman & Geman, 1984).** With logarithmic cooling $T_k = \frac{c}{\ln k}$, SA converges in probability to a global optimum.

2.3 Local Beam Search

Maintain a pool of K states; expand the neighbours of *all* and keep the best K. A *stochastic beam* variant retains states with probability proportional to fitness—essentially a “poor-man’s GA”.

2.4 Genetic Algorithms (GAs)

Population-based *evolutionary* local search. Key operators: **selection**, **crossover**, and **mutation**. GAs excel when the representation contains high-quality *building blocks* (schemata) that crossover can recombine effectively.

2.5 Tabu Search

Maintain a *tabu list* of recently visited states or moves to forbid cycling and promote exploration. An *aspiration* criterion allows overriding the tabu status if a move yields a new best-so-far solution.

2.6 Min-Conflicts (for CSPs)

At each step choose a conflicted variable and assign it the value that minimises the number of constraint violations. Empirically solves the N-Queens problem for $n=10^7$ in about 50 moves.

3 Landscape Phenomena

Table 3 highlights common topological features of search spaces and counter-measures.

Table 3: Typical landscape features and mitigation strategies

Feature	**Consequence**	**Mitigation**
Local Optima	Search stagnates	Random restarts, SA, Tabu
Plateaus	Flat region $\Rightarrow$ random walk	SA, sideway moves, larger neighbourhood
Ridges / Valleys	Steep walls restrict moves	2-Opt, variable-neighbourhood search
Deceptive Funnels	Good regions hidden behind bad ones	Diversification, hybrid algorithms

4 Continuous Local Search

When f is differentiable, *gradient-based* methods such as steepest descent, Newton's method, Momentum, or Adam can be interpreted as local search in $\mathbb{R}^n$. Key hyper-parameters include step size, learning-rate decay, and regularisation; ill-conditioning often necessitates preconditioning or adaptive methods.

5 Worked Examples

5.1 N–Queens via Min-Conflicts ($n = 8$)

1. **Initial state:** random queen positions.
2. Repeat until no conflicts:
   Pick a queen in conflict and *move* it to the row with the fewest conflicts. In practice, 10–20 moves usually suffice.

5.2 TSP with 2-Opt Hill-Climbing

1. **State** = permutation of cities.
2. **Neighbour** = choose indices $i < j$ and reverse the segment between them (2-Opt move).

3. While improvement exists, perform best-improving 2-Opt moves.
4. Optionally restart with SA to escape local minima.

Empirically, on random 100-city instances, 2-Opt + SA attains tour lengths within $\leq 5\%$ of the optimum.

6 Choosing the Right Method

Table 4: Heuristic guide: matching problem scenarios to local-search techniques

Scenario	**Suggested Technique**
Quick, small discrete instance	Random-Restart Hill-Climbing
Rugged landscape, unknown topology	Simulated Annealing
Many plateaus, short-term memory helpful	Tabu Search
Highly multimodal, modular solution space	Genetic Algorithms
Large CSP (scheduling, timetabling)	Min-Conflicts
Continuous, differentiable objective	Gradient-based with momentum / Adam

7 Implementation Tips

- **Efficient Δ-evaluation:** update the objective incrementally instead of recomputing from scratch.
- **Parameter tuning:** cooling rate, tabu tenure, population size, etc.
- **Hybridisation:** e.g. GA with local 2-Opt mutation (*memetic algorithm*).
- **Parallelism:** beam search and GAs are embarrassingly parallel; SA supports parallel tempering.
- **Termination:** fixed iterations, time budget, a no-improvement counter, or hitting a target value.

8 Advanced & Modern Variants

- **Variable Neighbourhood Search (VNS)**
- **Late-Acceptance Hill-Climbing** — compares to the cost k steps back.
- **Large Neighbourhood Search (LNS)** — destroy/repair strategy (widely used in CP-SAT and OR-Tools).
- **Simulated Quantum Annealing** — adiabatic quantum analogue.
- **Hyper-parameter Optimisation** — random search, Bayesian optimisation blend global and local moves.

9 Applications in AI

- **Planning & Scheduling** — job-shop, vehicle routing.
- **Computer Vision** — energy minimisation in MRFs (graph cuts $\approx$ large-scale local moves).
- **Robotics** — trajectory smoothing.
- **Machine Learning** — feature selection, architecture search, hyper-parameter tuning.
- **Games & Puzzles** — Sudoku, sliding-tile (IDA* uses RBFS $\approx$ local search in f-space).

10 Exercises

1. **Implement** Steepest-Ascent and SA on the 8-Queens problem; compare average step count vs. temperature schedule.
2. **Deadline Scheduling:** with weighted penalties, apply Tabu Search to minimise weighted tardiness; study tabu-tenure effects.
3. **GA Crossover Study:** on binary MAX-SAT, compare one-point vs. uniform crossover convergence.
4. **Theory:** prove that random-restart hill-climbing is *complete* in finite state spaces given non-zero probability of initialising in the global optimum basin.
5. **Research Survey:** summarise two recent hybrid meta-heuristics that combine reinforcement learning with local search.

11 Key Takeaways

- Local search trades strict *optimality guarantees* for *scalability*.
- Success hinges on thoughtful *representation* and *neighbourhood design*.
- Meta-heuristics (SA, tabu, GAs) supply *diversification* and *intensification* mechanisms.
- Hybrid methods dominate real-world optimisation practice.

12 Further Reading

1. Russell & Norvig, *Artificial Intelligence — A Modern Approach*, Chapters 4 and 5.
2. S. Kirkpatrick *et al.*, “Optimization by Simulated Annealing,” *Science*, 1983.
3. F. Glover & M. Laguna, *Tabu Search*.
4. J. Holland, *Adaptation in Natural and Artificial Systems* (Genetic Algorithms).
5. Hoos & Stützle, *Stochastic Local Search: Foundations and Applications*.