# Huristic Algorithms: Tabu Search Algorithm: An Advanced Meta-Heuristic Optimization Technique with memory

Advanced Algorithms
Department of Computer Science and Engineering

## 1. Introduction and Motivation

In previous discussions, we learned how heuristic optimization algorithms like **Hill Climbing** and **Simulated Annealing (SA)** help search for better solutions when an exhaustive search is computationally infeasible.

However, both these methods suffer from a key issue: *they may get trapped in local optima or revisit previously explored inferior solutions.*

To overcome this limitation, the **Tabu Search (TS)** algorithm introduces a form of *adaptive memory* that helps the search process avoid cycling and explore new regions of the solution space effectively.

## 2. Concept of Tabu Search

### 2.1 Core Idea

Tabu Search is an **iterative meta-heuristic** algorithm designed to improve upon local search methods by using memory structures.

It maintains a list of recently visited solutions or moves, called the **Tabu List**, which prevents the algorithm from revisiting them for a certain number of iterations (known as the **tabu tenure**).

This controlled memory mechanism allows the search to:

- Escape from local minima,
- Avoid cycling over previously explored solutions,
- Explore more diverse regions of the search space.

—

# 3. Basic Mechanism

**Tabu Search** enhances the basic local search (like hill climbing) as follows:

1. Start with an initial feasible solution $S$.
2. Generate a neighborhood $N(S)$ by making small modifications to $S$.
3. Choose the best neighbor $S'$ from $N(S)$, even if it is worse than $S$.
4. If $S'$ is not in the tabu list (i.e., not forbidden), move to $S'$.
5. Update the tabu list with the performed move or solution.
6. Optionally, override tabu restrictions using **Aspiration Criteria** if the move gives a better solution than any seen so far.
7. Repeat until a stopping criterion is met (like a maximum number of iterations or convergence).

—

# 4. Key Components of Tabu Search

## 4.1 Tabu List (Short-Term Memory)

The tabu list stores recently made moves (or visited solutions) to prevent immediate revisiting. The size of the list (tabu tenure) controls the memory length:

$$\text{Tabu Tenure} = k \text{ (number of iterations a move remains forbidden)}$$

## 4.2 Aspiration Criteria

Sometimes, a move may be tabu but still desirable (e.g., it leads to a global improvement). In such cases, we can override the tabu restriction using the aspiration rule:

$$\text{If } f(S') < f_{\text{best}}, \text{ then accept } S' \text{ even if tabu.}$$

—

# 5. Mathematical Formulation

Let:

$$S_t \text{ be the current solution at iteration } t$$

$$N(S_t) \text{ be the neighborhood of } S_t$$

We select:

$$S_{t+1} = \arg \min_{S' \in N(S_t) \setminus \text{Tabu}} f(S')$$

subject to aspiration criteria.

Here, $f(S)$ is the objective (or cost) function to be minimized.

—

# 6. Pseudocode for Tabu Search

```
TabuSearch()
  Initialize S = S0
  Best = S
  TabuList =
  while (stopping criterion not met)
      Generate neighborhood N(S)
      Select best S'  N(S) not in TabuList (or satisfying aspiration)
      S = S'
      if f(S) < f(Best)
          Best = S
      Update TabuList with recent move
      Remove oldest entries if TabuList too long
  end while
  return Best
```

—

# 7. Worked Example: Tabu Search on a 5-city TSP (Detailed Iterations)

We demonstrate Tabu Search on a small symmetric TSP with five cities $\{A, B, C, D, E\}$. We use *swap-two-cities* as the neighborhood operator and maintain a short-term tabu list that stores recent *swaps* (unordered pairs); tabu tenure is set to 3 iterations for each recorded swap. We also show an explicit aspiration-criterion scenario.

## 0.1 Distance matrix

| $d(\cdot,\cdot)$ | $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|---|
| $A$ | 0 | 2 | 9 | 10 | 7 |
| $B$ | 2 | 0 | 6 | 4 | 3 |
| $C$ | 9 | 6 | 0 | 8 | 5 |
| $D$ | 10 | 4 | 8 | 0 | 6 |
| $E$ | 7 | 3 | 5 | 6 | 0 |

The cost (energy) of a tour $S = [C_1, C_2, \ldots, C_n]$ is:

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

## 0.2 Initial solution

Choose an initial route (randomly or heuristically). Here:

$$S_0 = [A, B, C, D, E, A]$$

Compute its cost:

$$E(S_0) = d(A,B) + d(B,C) + d(C,D) + d(D,E) + d(E,A) = 2 + 6 + 8 + 6 + 7 = 29$$

Set:

$$\text{Current solution } S \leftarrow S_0, \quad \text{Best found } S_{\text{best}} \leftarrow S_0, \quad E_{\text{best}} \leftarrow 29$$

Tabu list initially empty.

## 0.3 Notation used in the steps

- We denote a swap (exchange of positions of cities $X$ and $Y$) by $(X, Y)$.
- Tabu list stores swaps (unordered pairs). Each stored entry has a remaining *tenure* (number of iterations until it expires).
- At each iteration we:
  1. Generate candidate neighbors by swapping every pair of cities (except the fixed start city if you fix it).
  2. Compute each neighbor's cost.
  3. Choose the best neighbor that is *not tabu* (or that satisfies aspiration).
  4. Update the tabu list: add the swap performed with tenure = 3, decrement all existing tenures, remove entries with tenure = 0.

## 0.4 Iteration table: candidates, costs and chosen move

We show a manageable subset of candidate swaps at each iteration (only swaps that produce interesting changes).

**Iteration 1 (start):**

$$S = [A, B, C, D, E, A], \quad E(S) = 29, \quad \text{TabuList} = \varnothing$$

Generate a few swaps and compute their costs:

- Swap $(B, E)$: $S' = [A, E, C, D, B, A]$. Cost:

  $$E(S') = d(A, E) + d(E, C) + d(C, D) + d(D, B) + d(B, A) = 7 + 5 + 8 + 4 + 2 = 26$$

- Swap $(C, D)$: $S' = [A, B, D, C, E, A]$. Cost:

  $$E(S') = 2 + 4 + 8 + 5 + 7 = 26$$

- Swap $(B, C)$: $S' = [A, C, B, D, E, A]$. Cost:

  $$E(S') = 9 + 6 + 4 + 6 + 7 = 32$$

- Swap $(D, E)$: $S' = [A, B, C, E, D, A]$. Cost:

  $$E(S') = 2 + 6 + 5 + 6 + 10 = 29$$

**Choice:** the best neighbor(s) give cost 26. We pick swap $(B, E)$ (tie-break arbitrarily).

$$S \leftarrow [A, E, C, D, B, A], \quad E(S) = 26$$

Update best:

$$S_{\text{best}} \leftarrow S, \quad E_{\text{best}} \leftarrow 26$$

Update tabu list: add $(B, E)$ with tenure=3.

$$\text{TabuList} = \{(B, E) : 3\}$$

**Iteration 2:**

$$S = [A, E, C, D, B, A], \quad E(S) = 26, \quad \text{TabuList} = \{(B, E) : 3\}$$

Consider candidate swaps (selected ones shown):

- Swap $(E, C)$: $S' = [A, C, E, D, B, A]$. Cost:

$$E(S') = 9 + 5 + 6 + 4 + 2 = 26$$

- Swap $(E, D)$: $S' = [A, D, C, E, B, A]$. Cost:

$$E(S') = 10 + 8 + 5 + 3 + 2 = 28$$

- Swap $(C, B)$: $S' = [A, E, B, D, C, A]$. Cost:

$$E(S') = 7 + 3 + 4 + 8 + 9 = 31$$

- Swap $(B, E)$: reversing the previous swap would produce $S_0$ of cost 29, but $(B, E)$ is *tabu* now (tenure=3). We therefore skip it unless aspiration applies.

**Choice:** best non-tabu neighbor has cost 26 (swap $(E, C)$). We may accept a non-improving equal-cost move to diversify.

$$S \leftarrow [A, C, E, D, B, A], \quad E(S) = 26$$

Update tabu: add swap $(E, C)$ with tenure=3; decrement existing tenures:

$$\text{TabuList} = \{(B, E) : 2,\ (E, C) : 3\}$$

**Iteration 3:**

$$S = [A, C, E, D, B, A], \quad E(S) = 26, \quad \text{TabuList} = \{(B, E) : 2, (E, C) : 3\}$$

Candidate swaps:

- Swap $(C, B)$: $S' = [A, B, E, D, C, A]$. Cost:

$$E(S') = 2 + 3 + 4 + 8 + 9 = 26$$

- Swap $(C, D)$: $S' = [A, D, E, C, B, A]$. Cost:

$$E(S') = 10 + 6 + 5 + 6 + 2 = 29$$

- Swap $(E, B)$: swapping (E,B) would return to an earlier configuration; $(B, E)$ is still tabu (tenure=2), so it is forbidden unless aspiration.

**Choice:** pick swap $(C, B)$ (cost 26). We accept it.

$$S \leftarrow [A, B, E, D, C, A], \quad E(S) = 26$$

Update tabu: add $(C, B)$ tenure=3; decrement others:

$$\text{TabuList} = \{(B, E) : 1,\ (E, C) : 2,\ (C, B) : 3\}$$

**Iteration 4:**

$$S = [A, B, E, D, C, A], \quad E(S) = 26, \quad \text{TabuList} = \{(B, E) : 1, (E, C) : 2, (C, B) : 3\}$$

Candidate swaps:

- Swap $(B, C)$: flipping back $(C, B)$ would be tabu (it is in the tabu list with tenure 3). The cost if we did it would be 26 (not an improvement over $E_{\text{best}} = 26$).
- Swap $(D, C)$: $S' = [A, B, E, C, D, A]$. Cost:

$$E(S') = 2 + 3 + 5 + 8 + 10 = 28$$

- Swap $(B, E)$: $(B, E)$ has tenure 1 (still tabu); if we executed it we'd get $S_0$ cost 29, which is worse.

No non-tabu neighbor improves $E_{\text{best}}$. According to Tabu Search policy, we choose the best *non-tabu* candidate (even if not improving). Suppose we choose swap $(D, C)$ giving cost 28.

$$S \leftarrow [A, B, E, C, D, A], \quad E(S) = 28$$

Update tabu: add $(D, C)$ with tenure=3; decrement existing tenures and remove those that expire:

$$\text{TabuList updates: } (B, E) : 0 \rightarrow \text{removed},\ (E, C) : 1,\ (C, B) : 2,\ (D, C) : 3$$

Final TabuList after iteration 4:

$$\text{TabuList} = \{(E, C) : 1,\ (C, B) : 2,\ (D, C) : 3\}$$

**Iteration 5:**

$$S = [A, B, E, C, D, A], \quad E(S) = 28, \quad E_{\text{best}} = 26$$

Tabu list: $(E, C) : 1,\ (C, B) : 2,\ (D, C) : 3$.

Consider candidates (examples):

- Swap $(B, C)$: $(C, B)$ is still tabu (tenure=2). If performed it would yield cost 26, which equals $E_{\text{best}}$ (not strictly better).

- Swap $(C, E)$: this swap currently has tenure 1 (tabu), and the resulting cost would be 26 (not better).
- Swap $(B, D)$: $S' = [A, D, E, C, B, A]$. Cost:

$$E(S') = 10 + 6 + 5 + 8 + 2 = 31$$

- Swap $(A, B)$: often we fix the start city in TSP implementations; if not, swapping the first city changes route arbitrarily.

No improving non-tabu move exists; pick best available non-tabu move (for demonstration we choose one with least deterioration). After the chosen swap we update the tabu list (decrement tenures, remove zeros) and continue.

---

## 0.5 Aspiration criterion — explicit numerical scenario

The aspiration criterion is an override rule: *even if a move is tabu*, if it yields a solution better than any found so far (i.e., $E(S') < E_{\text{best}}$), we accept it.

We now show a concrete numerical aspiration example (this is separate from the main sequence above):

- Suppose at some iteration the current best cost is $E_{\text{best}} = 26$.
- A candidate move $m = (X, Y)$ is **tabu** (present in the tabu list).
- The neighbor $S'$ produced by applying $m$ yields $E(S') = 23$.
- Because $23 < E_{\text{best}} = 26$, the aspiration criterion allows the tabu move:

$$\text{Even though } m \in \text{TabuList}, \text{ accept } S' \text{ because } E(S') < E_{\text{best}}.$$

- We then update:

$$S \leftarrow S', \quad E_{\text{best}} \leftarrow 23, \quad S_{\text{best}} \leftarrow S'$$

  and still record the move in the tabu list (it may remain tabu for some iterations, but aspiration has allowed the improvement).

This demonstrates the *intended* behavior of aspiration: tabu protects the search from cycling, but not at the cost of missing major improvements.

### 0.6 How tabu entries are stored (practical notes)

- **Representation:** store unordered swap pairs (e.g., record the pair $\{B, E\}$), or store move descriptors such as "swap positions 2 and 5". Unordered pairs are often simpler for symmetric TSP.
- **Tabu tenure:** a small integer (3–7) is common; it determines how many iterations the move remains forbidden.
- **Updating:** each iteration:
  1. Decrement the tenure of every entry in the tabu list.
  2. Remove entries with tenure = 0.
  3. Add the new performed move with its initial tenure.
- **Aspiration check:** when selecting the best candidate, if the best candidate is tabu, check if it satisfies aspiration (e.g., yields $E < E_{\text{best}}$). If yes, accept despite tabu; if no, consider next best non-tabu candidate.

# 8. Comparison with Other Algorithms

| Feature | Hill Climbing / SA | Tabu Search |
|---|---|---|
| Memory | None or implicit (SA) | Explicit memory (tabu list) |
| Escape from Local Minima | Probabilistic (SA) | Deterministic via tabu rules |
| Best Use Case | Noisy landscapes | Structured combinatorial optimization |
| Parameter Control | Temperature (SA) | Tabu tenure, aspiration, diversification |

—

# 9. Advantages and Limitations

## Advantages

- Avoids cycling and local minima traps.
- Systematic exploration using memory.
- Flexible and adaptable to many problem domains.

### Limitations

- Computationally expensive due to memory updates.
- Sensitive to tabu tenure length.
- Performance depends on neighborhood design.

## 10. Exploration vs. Exploitation

Metaheuristic algorithms must balance two conflicting objectives:

- **Exploration:** Searching new or less-visited regions of the solution space to avoid local optima.
- **Exploitation:** Refining the best solutions found so far to achieve local improvement.

Too much exploitation can cause the algorithm to get trapped in a local minimum, while too much exploration can prevent convergence. A good optimization strategy gradually transitions from exploration to exploitation.

**Example:** In the Traveling Salesman Problem (TSP), large random swaps between distant cities promote exploration, while small local swaps around the best route promote exploitation.

**In Metaheuristics:**

| Algorithm | Exploration Mechanism | Exploitation Mechanism |
|---|---|---|
| Simulated Annealing | High temperature, random jumps | Low temperature, local improvements |
| Tabu Search | Tabu list to diversify search | Aspiration and local search refinement |
| Genetic Algorithm | Mutation and diversity | Selection and crossover |

—

# 11. Future Learning and Research Directions

- **Hybrid Approaches:** Combining Tabu Search with Simulated Annealing or Genetic Algorithms to balance exploration and exploitation.
- **Adaptive Tabu Tenure:** Dynamically adjust tabu length based on search progress.
- **Parallel Tabu Search:** Distribute search across multiple processors for large-scale problems.

- **Applications:** Scheduling, routing, job-shop optimization, and VLSI design.

—