中山大學
SUN YAT-SEN UNIVERSITY

Lecture 13 Optimization Algorithms (II)

Algorithm Design

zhangzizhen@gmail.com

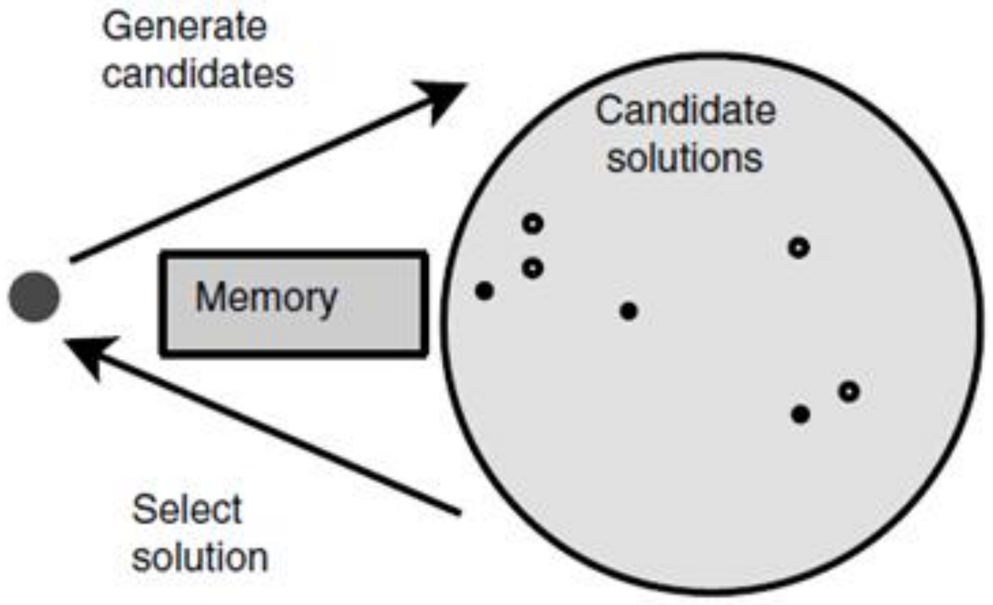
QQ group: 117282780

Single-Solution Based Metaheuristics

- Common Concepts
- Local Search
- Simulated Annealing
- Tabu Search
- Iterated Local Search
- Variable Neighborhood Search
- GRASP

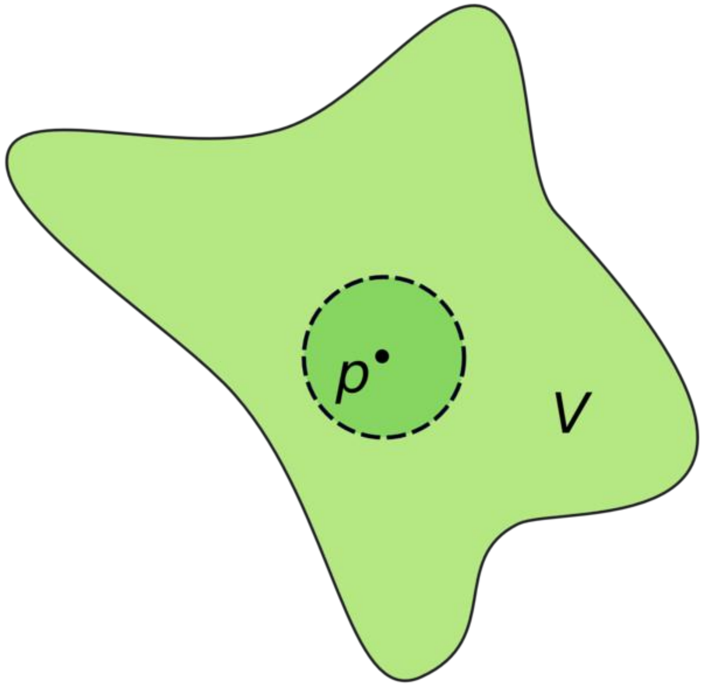
Common Concepts

- Single-metaheuristics iteratively apply the ***generation*** and ***replacement*** procedure from the current single solution.



Common Concepts

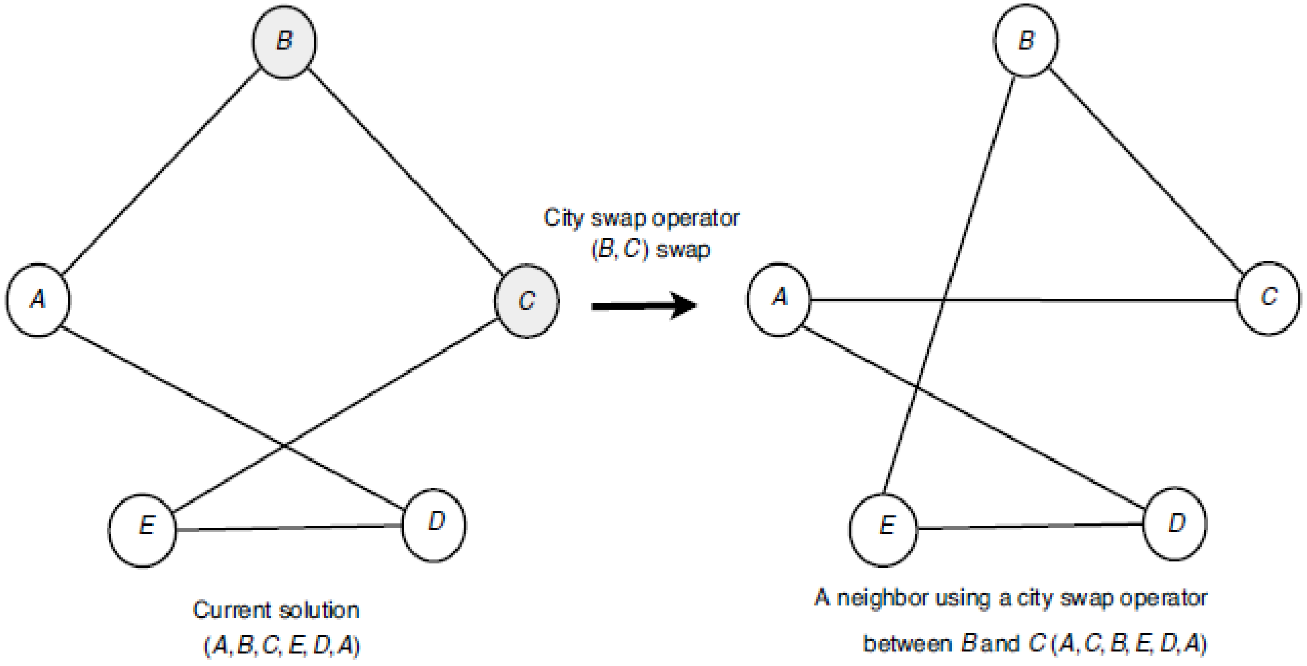
- **Neighborhood**
  - plays a crucial role in the performance of a single-metaheuristic.



- A solution in the neighborhood is called a ***neighbor***.
- A neighbor *s'* is generated by modifying the current solution *s*.
- The area of the neighborhood is relied on the ***operator*** employed. (operators can be regarded the ways or rules of modifying *s*.)

Neighborhood Operators

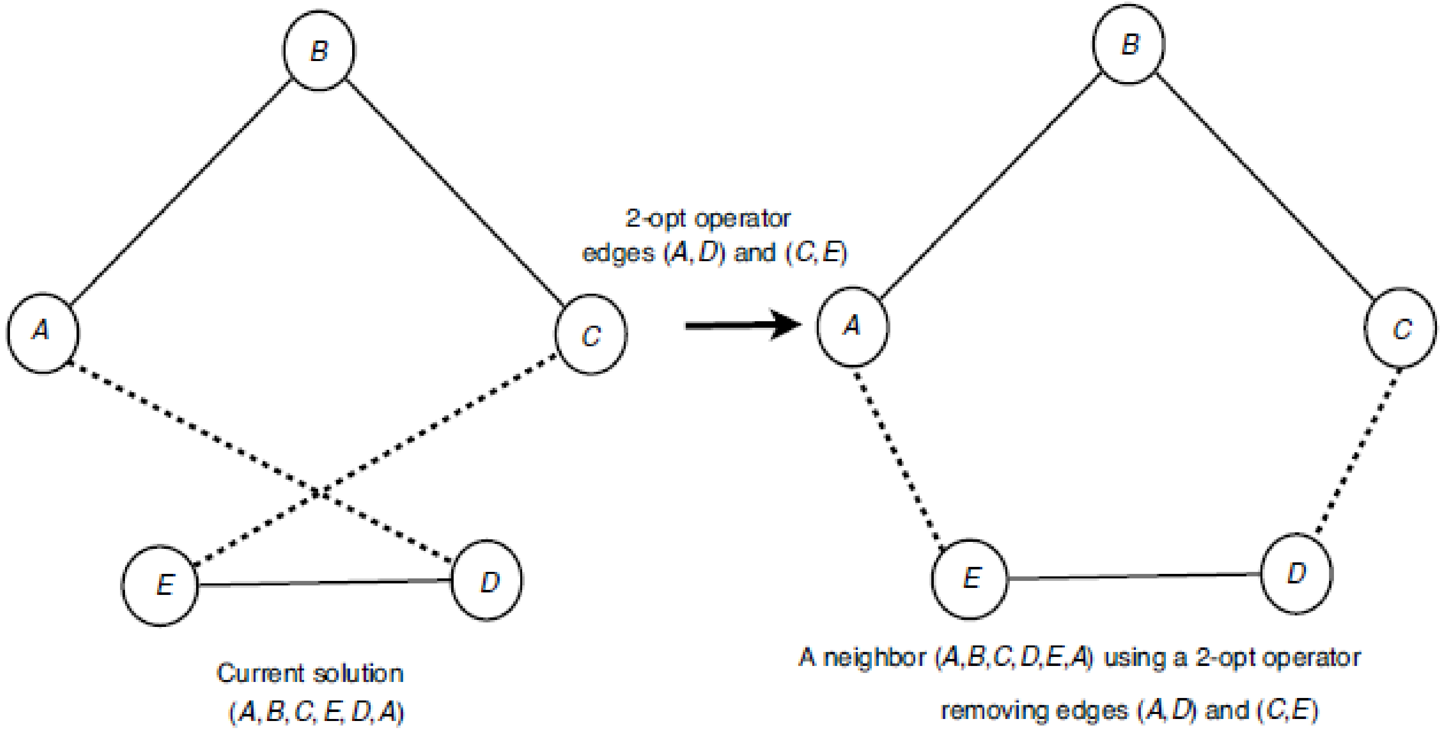
- For permutation problems, such as the TSP, single machine scheduling problem and *N* queens problem, the **exchange operator** (swap operator) may be used.



The size of this neighborhood is *n*(*n*-1)/2, where *n* is the number of cities.

Neighborhood Operators

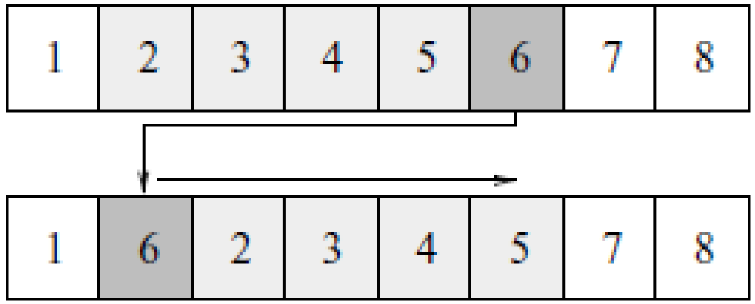
- **2-opt operator**


B
A
C
E
D
Current solution
(A,B,C,E,D,A)
2-opt operator
edges (A,D) and (C,E)
B
A
C
E
D
A neighbor (A,B,C,D,E,A) using a 2-opt operator
removing edges (A,D) and (C,E)

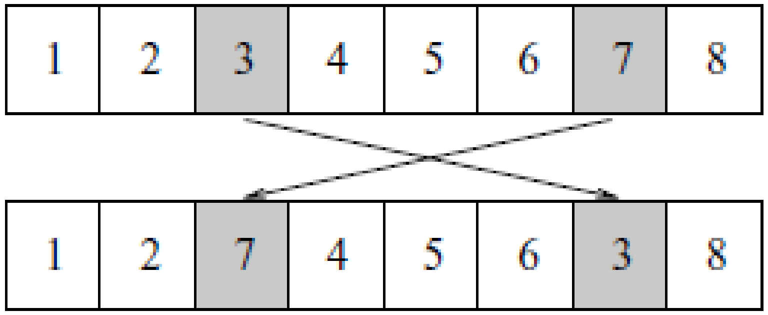
The size of the neighborhood for the 2-opt operator is [(*n*(*n*-1)/2) - *n*]; All pairs of edges are concerned except the adjacent pairs.

Neighborhood Operators

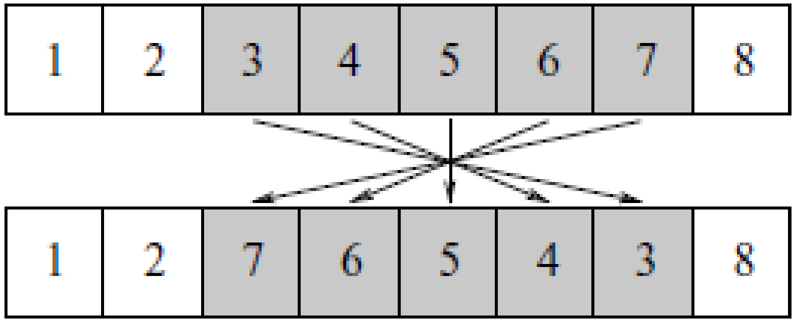
Insertion operator



Exchange operator



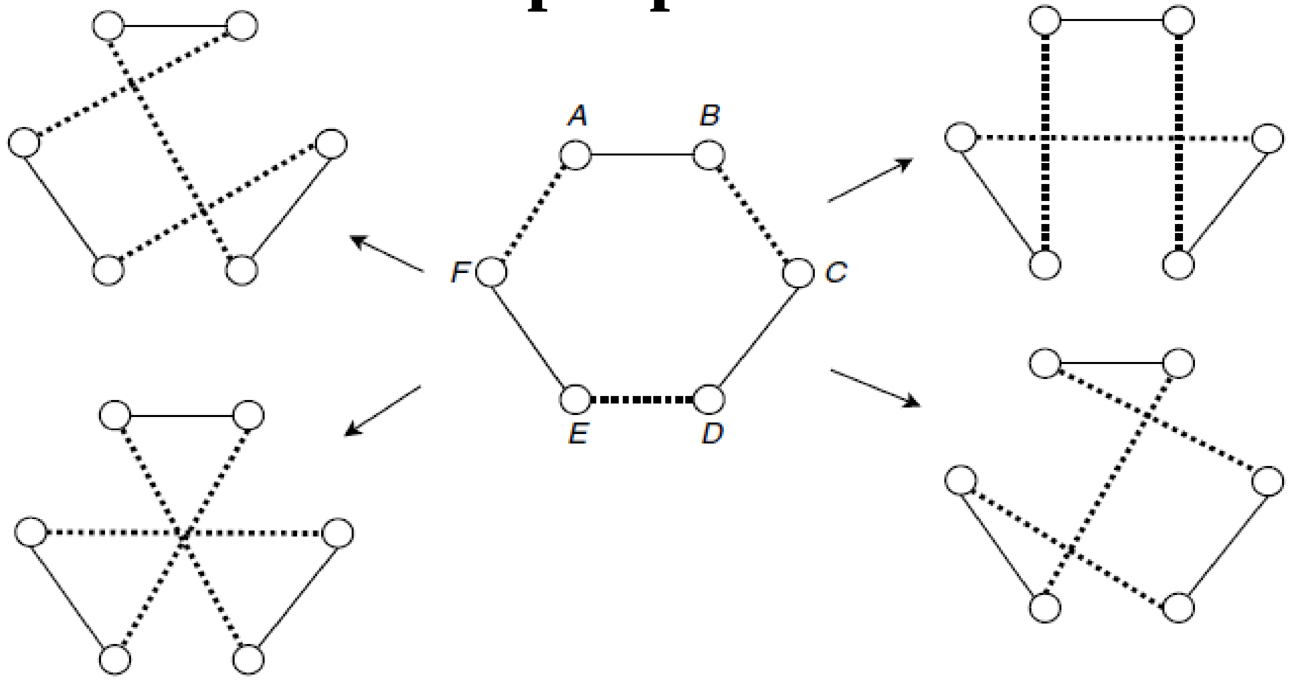
Inversion operator



Neighborhood Operators

- Another widely used operator is the *k*-opt operator, where *k* edges are removed from the solution and replaced with other *k* edges.
- The time complexity for 2-opt, 3-opt and 4-opt is $O(n^2)$, $O(n^3)$ and $O(n^4)$.

3-Opt operator



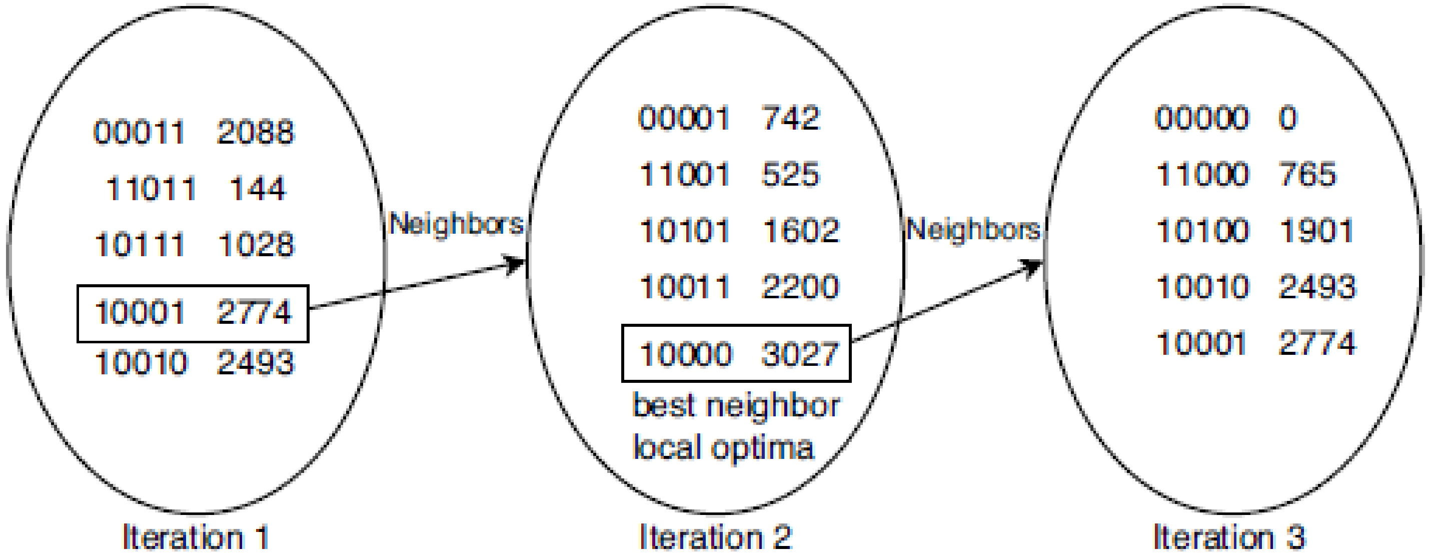
3-opt operator for the TSP. The neighbors of the solution (A,B,C,D,E,F) are (A,B,F,E,C,D), (A,B,D,C,F,E), (A,B,E,F,C,D), and (A,B,E,F,D,C).

Local Search (局部搜索)

- It is also called *hill climbing*, *descent*, *iterative improvement*, and so on.
- It is likely the oldest and simplest metaheuristic method.
- It starts at a given initial solution.
- At each iteration, the heuristic ***replace***s the current solution by a neighbor that ***improve***s the objective function.
- It stops when all candidate neighbors are worse than the current solution, i.e., a local minimum is reached.

LS Example

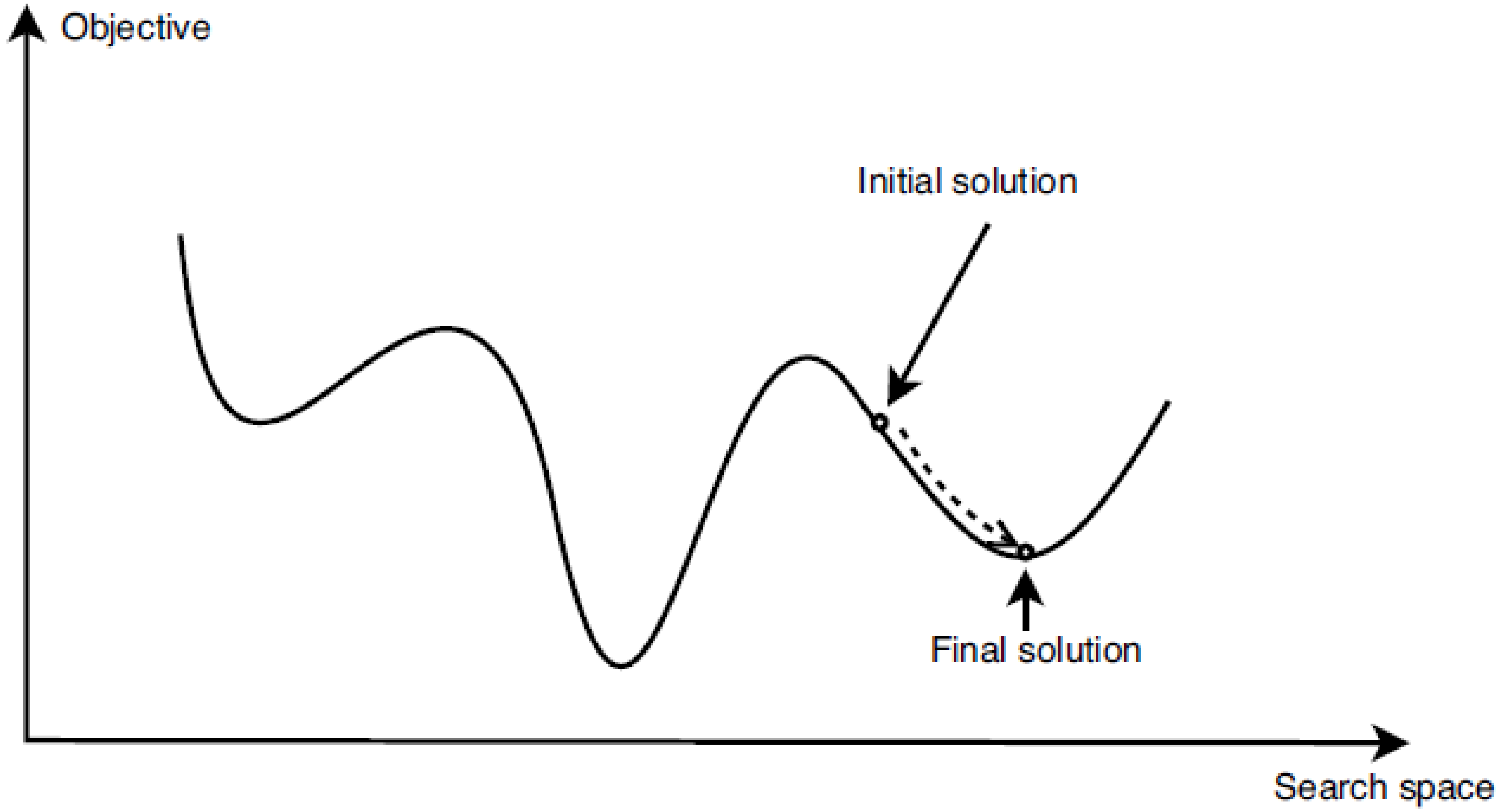
- Maximize $x^3 - 60x^2 + 900x$, x is discrete



- Local search process using a binary representation of solutions, a flip move operator, and the best neighbor selection strategy.
- The global optimal solution is f($[01010]_2$) = f(10) = 4000, while the final local optimal found is s = [10000], starting from the solution s0=[10001)]

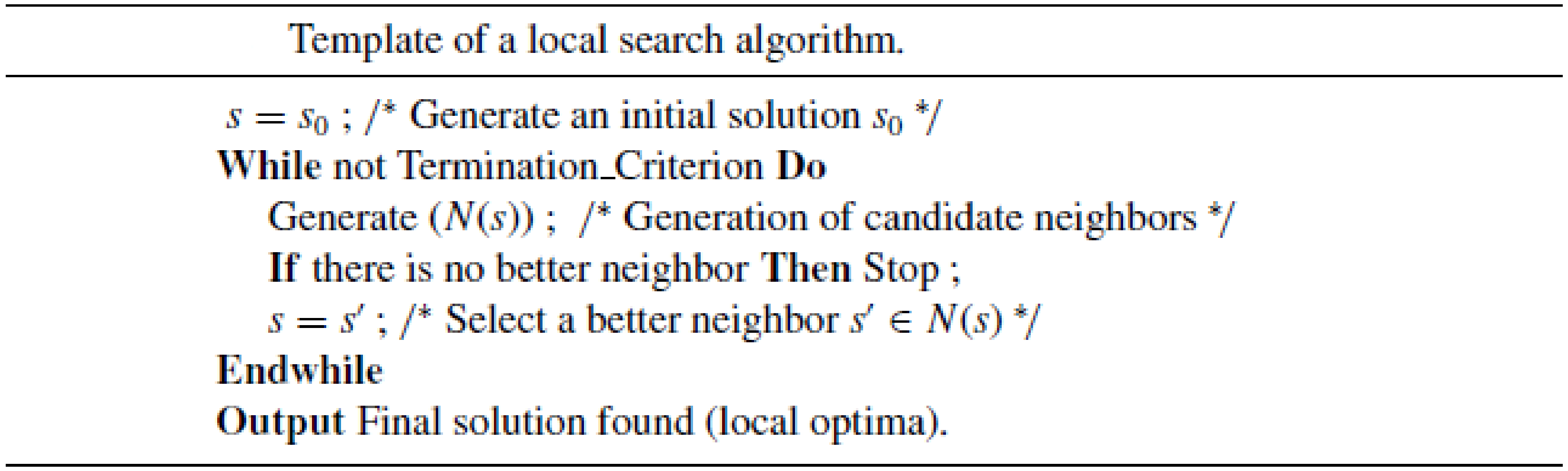
Questions

- How to generate a set of neighbors?
- How to select a neighbor?



How LS Works

- LS may be seen as a descent walk in the graph G=(S, V) representing the search space.
  - S represents the set of all feasible solutions.
  - V represents the neighborhood relation.
  - Each edge (i, j) in the graph will connect any neighboring s_i and s_j.
  - For a given solution s, the number of associated edges will be $|N(s)|$.

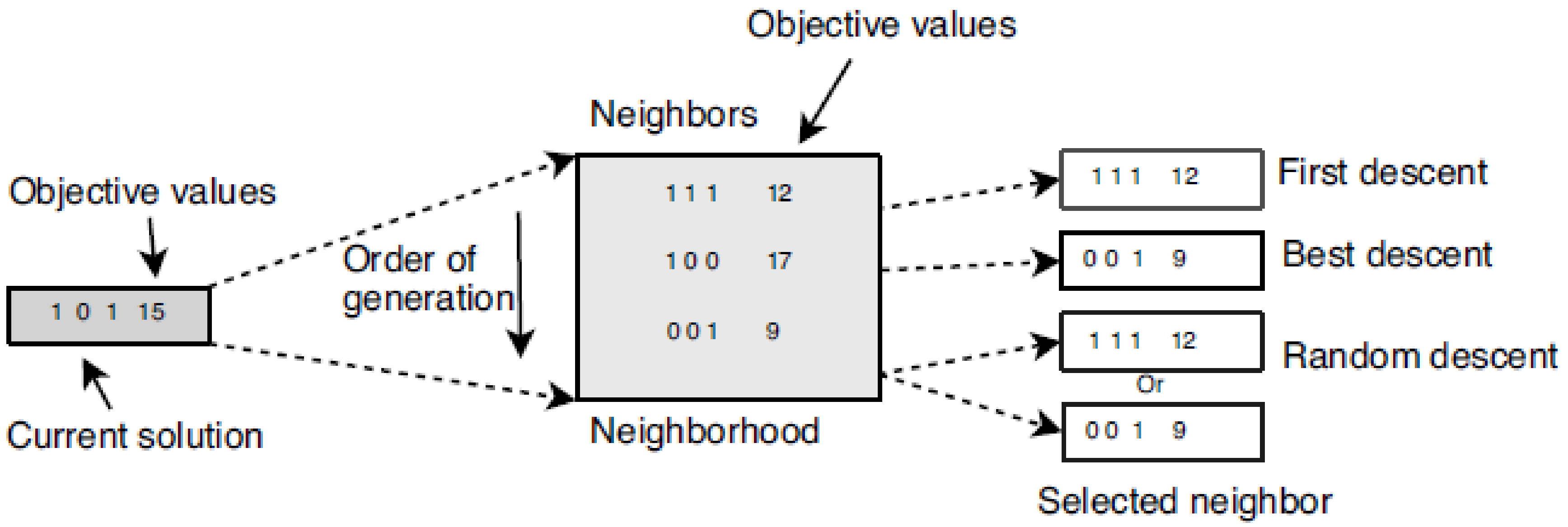


Template of a local search algorithm.

```
$s = s_0$ ; /* Generate an initial solution $s_0$ */
While not Termination_Criterion Do
    Generate ($N(s)$) ;  /* Generation of candidate neighbors */
    If there is no better neighbor Then Stop ;
    $s = s'$ ; /* Select a better neighbor $s' \in N(s)$ */
Endwhile
Output Final solution found (local optima).
```

How LS Works

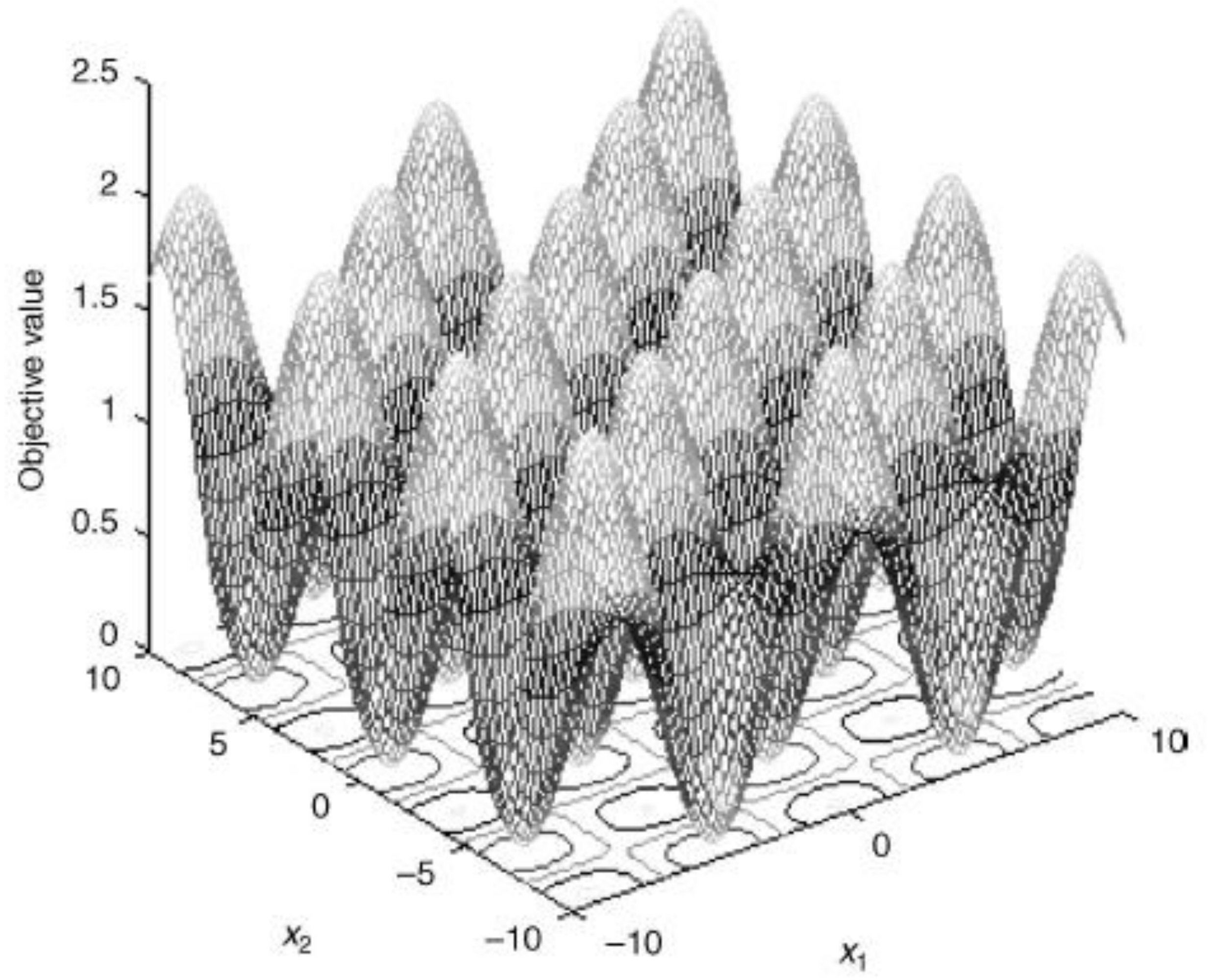
- **Selection of the Neighbor**
  - Best improvement (steepest descent)
  - First improvement
  - Random selection



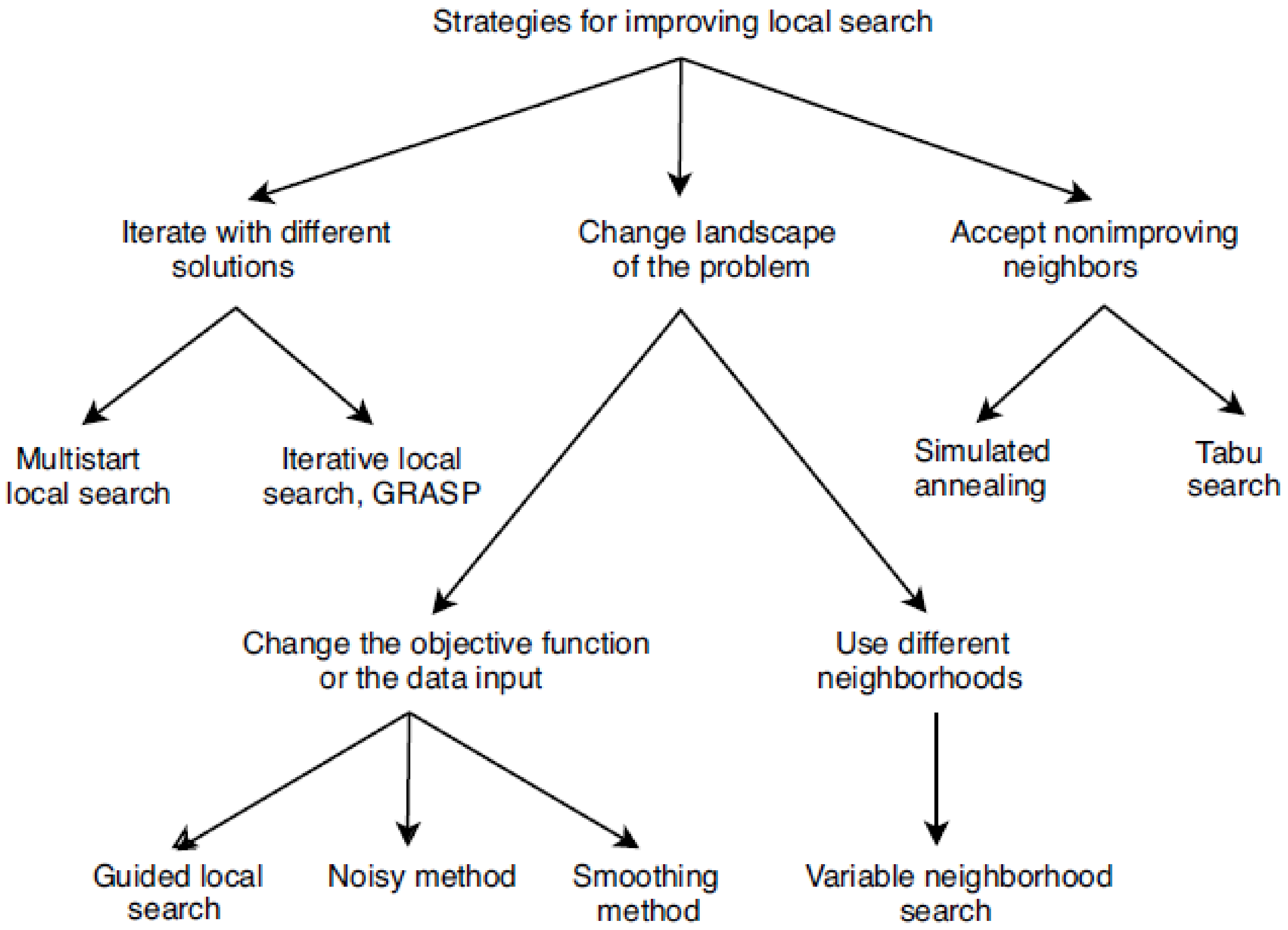
How LS Works

- Escaping from Local Optima
  - The LS is very sensitive to the initial solution.
  - No means to estimate the gap between the local optimum and the global optimum.
  - The number of iterations performed may not be known in advance.
  - Even if the LS runs very quickly, its worst case complexity is *exponential*.
  - Local search works well if there are not too many local optima.

Highly Multimodal Function

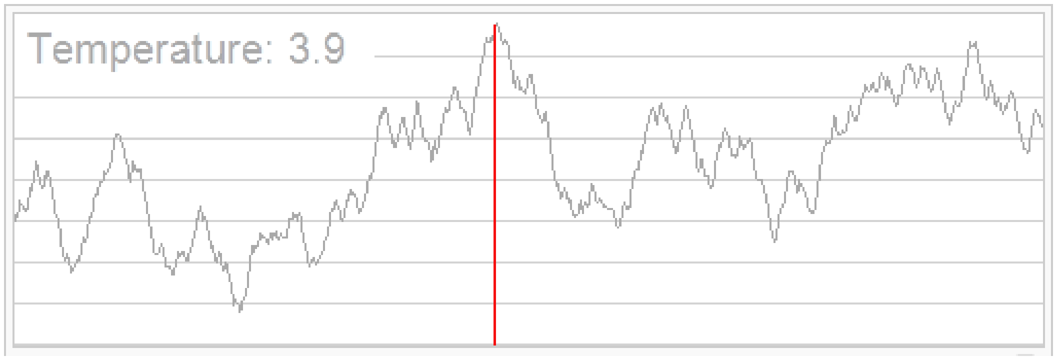


How to avoid local optima


Strategies for improving local search
Iterate with different solutions
Change landscape of the problem
Accept nonimproving neighbors
Multistart local search
Iterative local search, GRASP
Simulated annealing
Tabu search
Change the objective function or the data input
Use different neighborhoods
Guided local search
Noisy method
Smoothing method
Variable neighborhood search

Simulated Annealing (SA)

- In the pioneering works, SA has been applied to graph partitioning and VLSI design.
- Simple and efficient in solving combinatorial optimization problems.
- It has been extended to deal with continuous optimization problems.
- SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure.



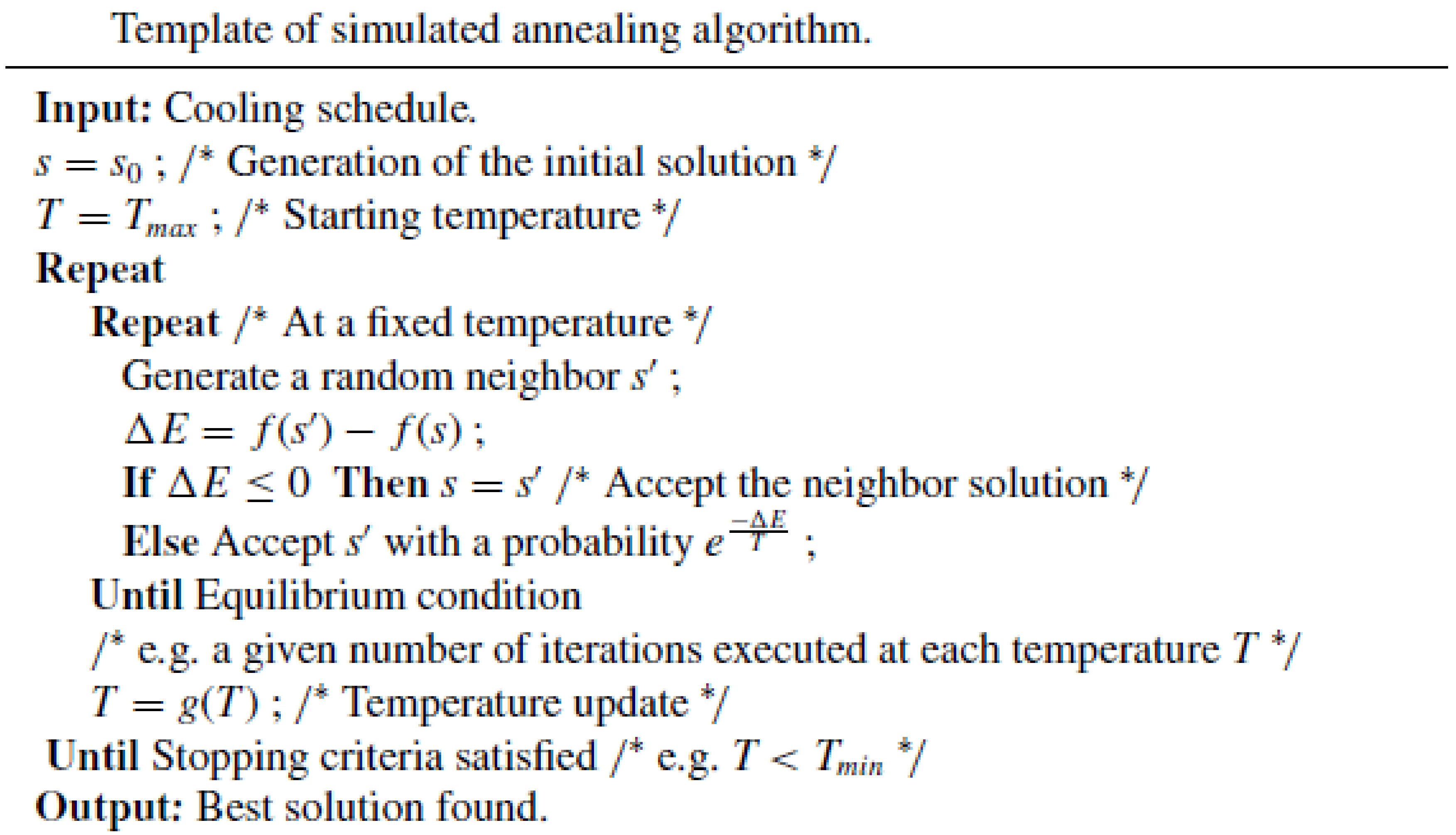
Description of SA

- At each iteration, a random neighbor s' is generated.
- Moves that improve the cost function are always accepted.
- Otherwise, the neighbor is selected with a given probability: (important!)

$$P(\Delta E, T) = e^{-\frac{f(s')-f(s)}{T}}$$

- Temperature *T* determines the probability of accepting non-improving solutions (How?).
- At a particular level of temperature, many trials are explored. Once an equilibrium state (what is this?) is reached, the temperature is gradually decreased according to a cooling schedule (why do so?).

SA Algorithm



Template of simulated annealing algorithm.

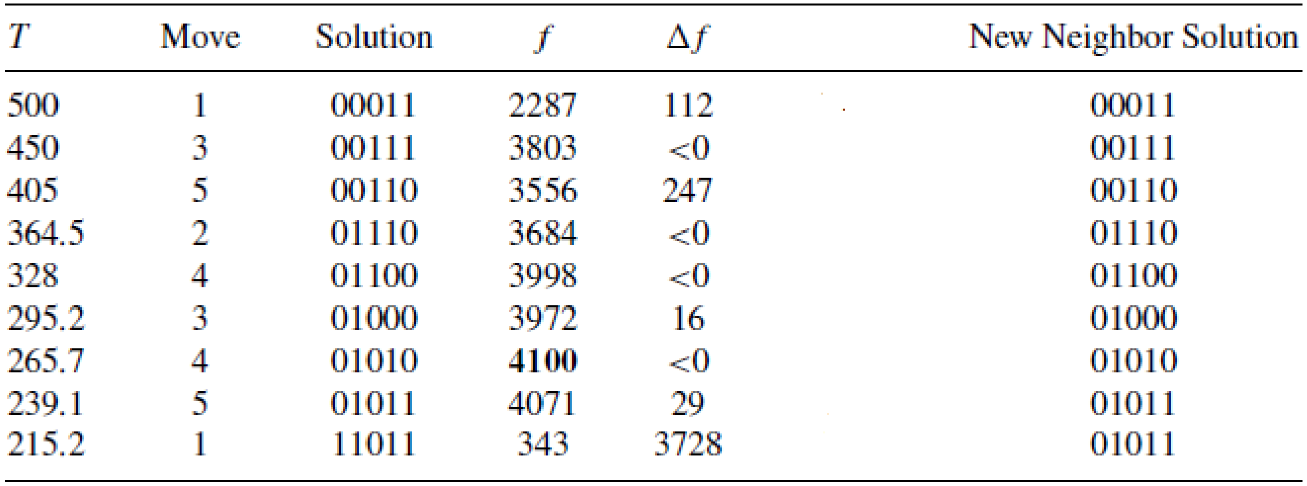
```
Input: Cooling schedule.
$s = s_0$ ; /* Generation of the initial solution */
$T = T_{max}$ ; /* Starting temperature */
Repeat
    Repeat /* At a fixed temperature */
      Generate a random neighbor $s'$ ;
      $\Delta E = f(s') - f(s)$ ;
      If $\Delta E \leq 0$ Then $s = s'$ /* Accept the neighbor solution */
      Else Accept $s'$ with a probability $e^{\frac{-\Delta E}{T}}$ ;
    Until Equilibrium condition
    /* e.g. a given number of iterations executed at each temperature $T$ */
    $T = g(T)$ ; /* Temperature update */
 Until Stopping criteria satisfied /* e.g. $T < T_{min}$ */
Output: Best solution found.
```

SA Example

- Maximize $f(x) = x^3 - 60x^2 + 900x + 100$, where x is discrete.
- A solution is represented as a string of 5 bits.
- The global maximum of this function is 01010 ($x = 10$, $f(x) = 4100$).
- The first scenario starts from the solution 10011 ($x=19$, $f(x) = 2399$) with an initial temperature T_0 equal to 500.
- The second scenario starts from the same solution 10011 with an initial temperature T_0 equal to 100.
- The initial temperature is not high enough and the algorithm gets stuck by local optima.

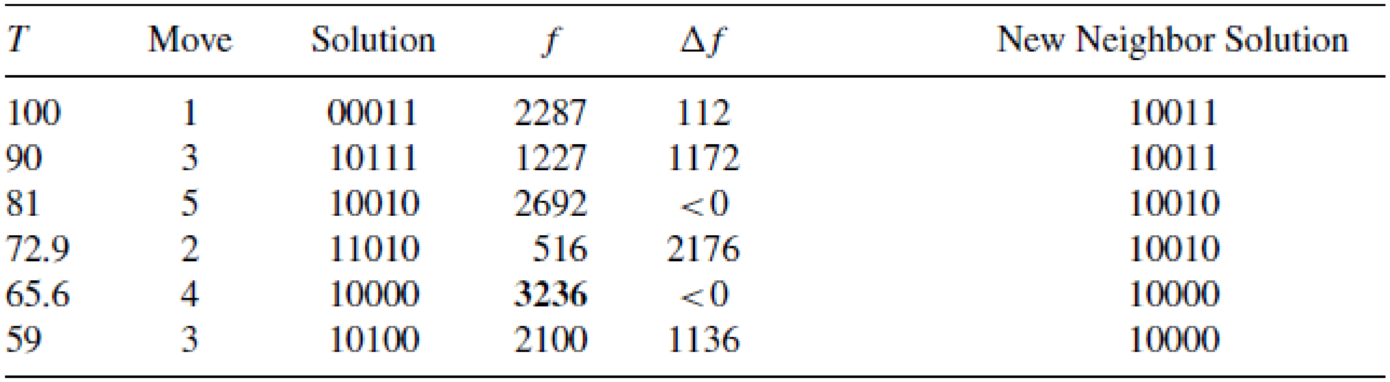
SA Example

First Scenario $T = 500$ and Initial Solution (10011)



T	Move	Solution	f	Δf	New Neighbor Solution
500	1	00011	2287	112	00011
450	3	00111	3803	<0	00111
405	5	00110	3556	247	00110
364.5	2	01110	3684	<0	01110
328	4	01100	3998	<0	01100
295.2	3	01000	3972	16	01000
265.7	4	01010	**4100**	<0	01010
239.1	5	01011	4071	29	01011
215.2	1	11011	343	3728	01011

Second Scenario: T = 100 and Initial Solution (10011). When Temperature is not High Enough, Algorithm Gets Stuck



T	Move	Solution	f	Δf	New Neighbor Solution
100	1	00011	2287	112	10011
90	3	10111	1227	1172	10011
81	5	10010	2692	<0	10010
72.9	2	11010	516	2176	10010
65.6	4	10000	**3236**	<0	10000
59	3	10100	2100	1136	10000

Move Acceptance

- The system can escape from local optima due to the probabilistic acceptance of a non-improving neighbor.
- At high temperature, the probability of accepting worse moves is high (Why?).
- If $T = +\infty$, all moves are accepted, which corresponds to a random walk in the feasible region.
- If $T = 0$, no worse moves are accepted and the search is equivalent to local search.

Equilibrium State

- To reach an equilibrium state at each temperature, a number ($N_{nonimprov}$) of non-improving iterations must be performed.
- This number must be set according to the size of the problem instance and particularly proportional to the neighborhood size $|N(s)|$.
- This number may be set by the following two ways:
  - **Static**: this number is determined before the search starts.
  - **Adaptive**: This number will be adjusted during the search process.

Cooling

- The temperature is decreased gradually such that

  $T_i > 0, \forall i$ and $\lim_{i \to +\infty} T_i = 0$.
- If the temperature is decreased slowly, better solutions are obtained but with more computation time.
- The temperature T can be updated in different ways:
  - **Linear**: $T = T - \beta$, where β is a specific constant value. Hence, we have $T_i = T_0 - i \times \beta$.
  - **Geometric**: $T = \alpha T$, where $\alpha \in [0, 1]$. It is the most popular cooling function. Experience has shown that α should be between 0.5 and 0.99.
  - **Adaptive**: In an adaptive cooling schedule, the decreasing rate is dynamic and depends on some information obtained during the search.

Stopping Condition

- Reaching a final temperature T_F which is the most popular stopping criteria. This temperature must be low (e.g., $T_{\text{min}} = 0.01$).
- Achieving a predetermined number of iterations without improving the best found solution.

Thank you!