Metaheuristic

Heuristics

- Greek verb: Heuriskein(to find, to discover)
- Archimedes is said to have run naked down the street shouting “Heureka” (I have found it) after discovering the principle of flotation in his bath.
- The study of methods for discovering and inventing problem-solving techniques with mathematical proofs. (George Polya, *How to solve it*)
- Opposite of algorithmic
- Generate very good solutions without exhaustive search

Metaheuristic

- a higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to fit a particular kind of problem, especially with incomplete or imperfect information or limited computation capacity.
- a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality(fitness function).
- makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions.
- does not guarantee an optimal solution is ever found. Many metaheuristics implement some form of stochastic optimization.

Mathematics algorithm	Heuristic	Metaheuristic
Linear programming Integer programming	Greedy Search	TABU Search
optimum solution	near optimum solution	near optimum solution
No efficiency in NP-hard	efficiency , but easily find local optimum	efficiency and move to global optimum

Table of metaheuristics

Name	Abbreviation	Main category	Subcategory	Year published	Ref.
Simulated Annealing	SA	Trajectory-based	-	1983	[1]
Tabu Search	TS	Trajectory-based	-	1989	[2]
Genetic Algorithm	GA	Evolutionary-based	-	1992	
Evolutionary Algorithm	EA	Evolutionary-based	-	1994	
Cultural Algorithm	CA			1994	[3]
Particle Swarm Optimization	PSO	Nature-inspired	Swarm-based	1995	[4]
Differential Evaluation	DE	Evolutionary-based	-	1997	[5]

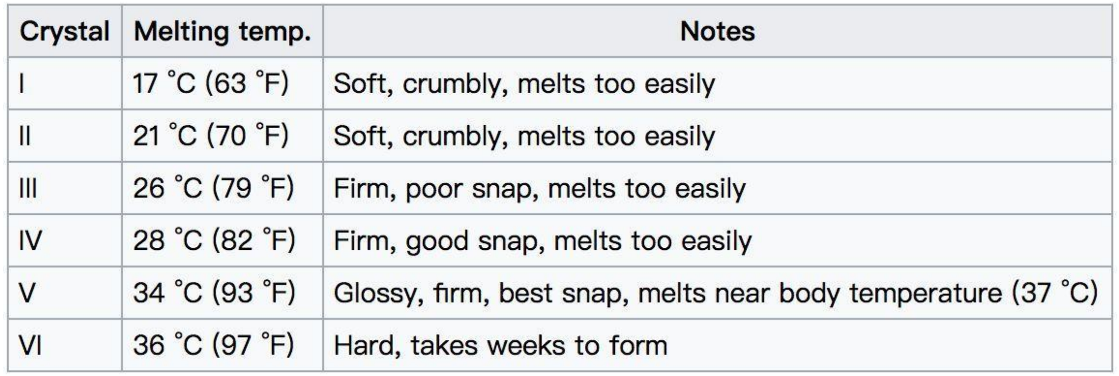
Local Search	LS			1997	
Variable Neighborhood Search	VNS	Trajectory-based	-	1997	[6]
Guided Local Search	GLS	Trajectory-based	-	1998	[7]
Clonal Selection Algorithm	CSA	Evolutionary-based	-	2000	[8]
Harmony Search	HS	Evolutionary-based	-	2001	[9]
Memetic Algorithm	MA	Evolutionary-based	-	2002	
Iterative Local Search	ILS	Trajectory-based	-	2003	[10]
Artificial Bee Colony	ABC	Nature-inspired	Bio-inspired	2005	[11]
Ant Colony Optimization	ACO	Nature-inspired	Bio-inspired	2006	[12]
Glowworm Swarm Optimization	GSO	Nature-inspired	Swarm-based	2006	[13]
Shuffled Frog Leaping Algorithm	SFLA	Nature-inspired	Bio-inspired	2006	[14]
Invasive Weed Optimization	IWO	Nature-inspired	Plant-based	2006	[15]
Imperialistic Competitive Algorithm	ICA	Nature-inspired	Human-based	2007	[16]

More: https://en.wikipedia.org/wiki/Table_of_metaheuristics

Simulated Annealing

- an optimization method which mimics the slow cooling of metals
- characterized by a progressive reduction in the atomic movements that reduce the density of lattice defects until a lowest-energy state is reached
- at each virtual annealing temperature, the simulated annealing algorithm generates a new potential solution (or neighbor of the current state) to the problem considered by altering the current state, according to a predefined criterion
- The acceptance of the new state is based on the satisfaction of the Metropolis criterion, and this procedure is iterated until convergence.

https://www.sciencedirect.com/topics/engineering/simulated-annealing-algorithm



Crystal	Melting temp.	Notes
I	17 °C (63 °F)	Soft, crumbly, melts too easily
II	21 °C (70 °F)	Soft, crumbly, melts too easily
III	26 °C (79 °F)	Firm, poor snap, melts too easily
IV	28 °C (82 °F)	Firm, good snap, melts too easily
V	34 °C (93 °F)	Glossy, firm, best snap, melts near body temperature (37 °C)
VI	36 °C (97 °F)	Hard, takes weeks to form

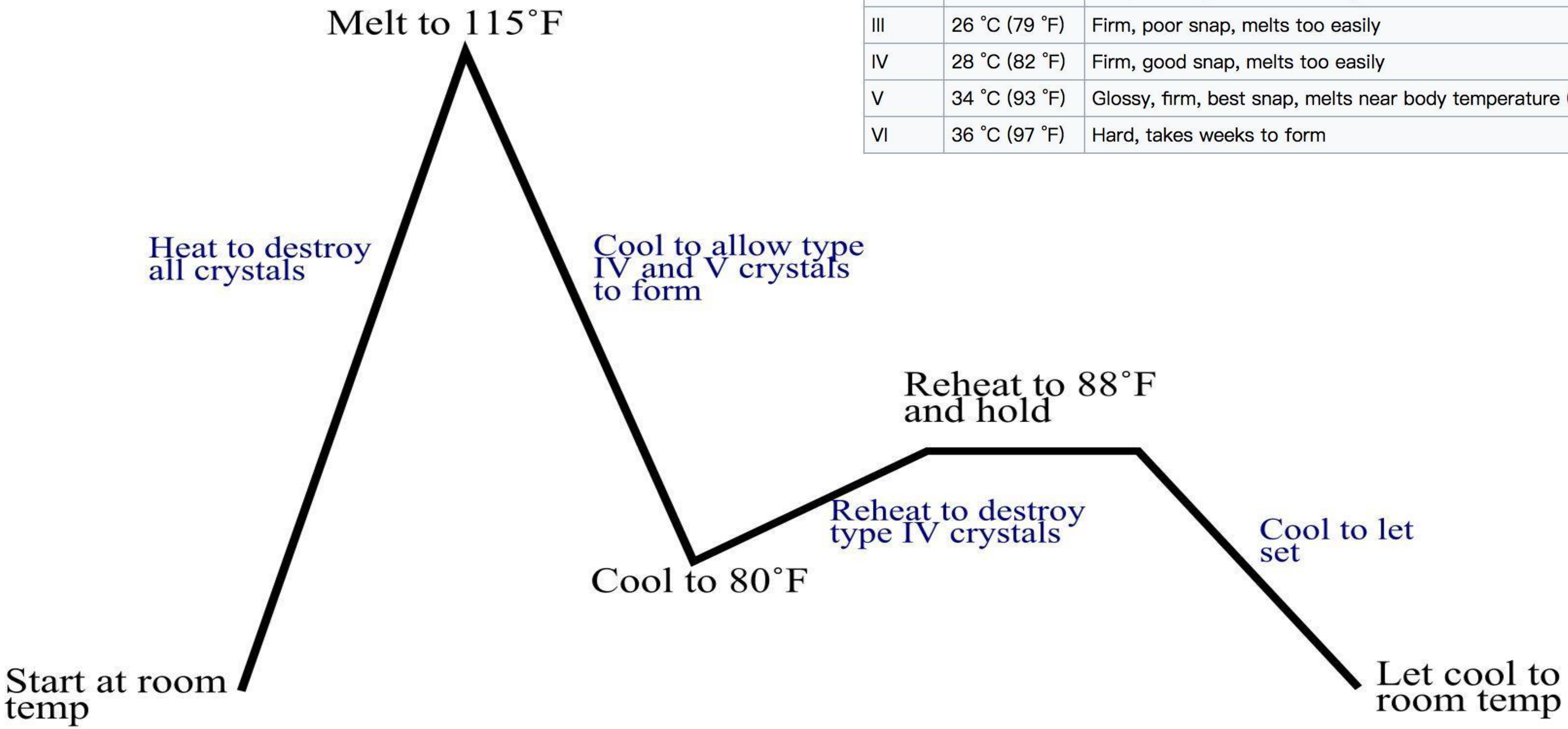

Melt to 115°F
Heat to destroy all crystals
Cool to allow type IV and V crystals to form
Reheat to 88°F and hold
Reheat to destroy type IV crystals
Cool to let set
Cool to 80°F
Start at room temp
Let cool to room temp

Diagram depicting the tempering process for milk chocolate.

What Is Simulated Annealing?

–applied to solve optimization problems

–a stochastic algorithm

–escaping from local optima by allowing worsening moves

–a memoryless algorithm, the algorithm does not use any information gathered during the search

–applied for both combinatorial and continuous optimization problems

–simple and easy to implement.

–motivated by the physical annealing process

Real Annealing Technique

Annealing Technique is known as a thermal process for **obtaining low-energy state** of a solid in a heat bath.

The process consists of the following two steps:

– **Increasing temperature**: Increase the **temperature** of the heat bath to a maximum value at which the solid melts.

– **Decreasing temperature:** Decrease carefully the temperature of the heat bath until the **particles** arrange themselves in the **ground state** of the solid.

- In the **liquid phase** all **particles** arrange themselves randomly, whereas in the ground state of the solid, the particles are arranged in a highly structured lattice, for which the corresponding energy is minimal.
- The **ground state** of the solid is obtained only if:

  the maximum value of the temperature is sufficiently high and the cooling is done sufficiently slow.
- Strong solid are grown from careful and slow cooling.

Metastable states

– If the initial temperature is not sufficiently high or a fast cooling is applied, **metastable states** (imperfections) are obtained.

Quenching

– The process that leads to metastable states is called q**uenching**

Thermal equilibrium

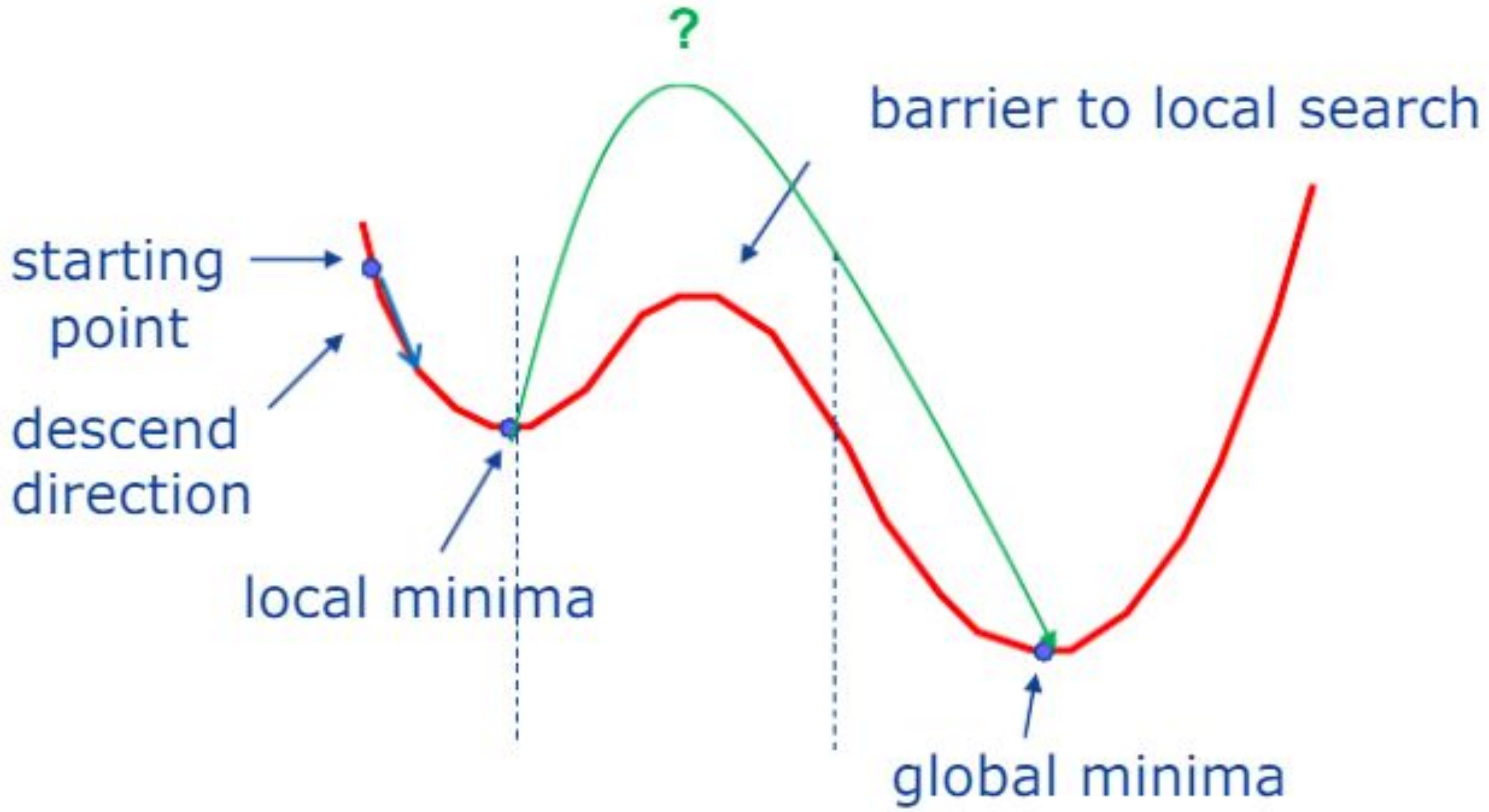
– If the **lowering of the temperature** is done sufficiently slow, the solid can reach **thermal equilibrium** at each temperature.

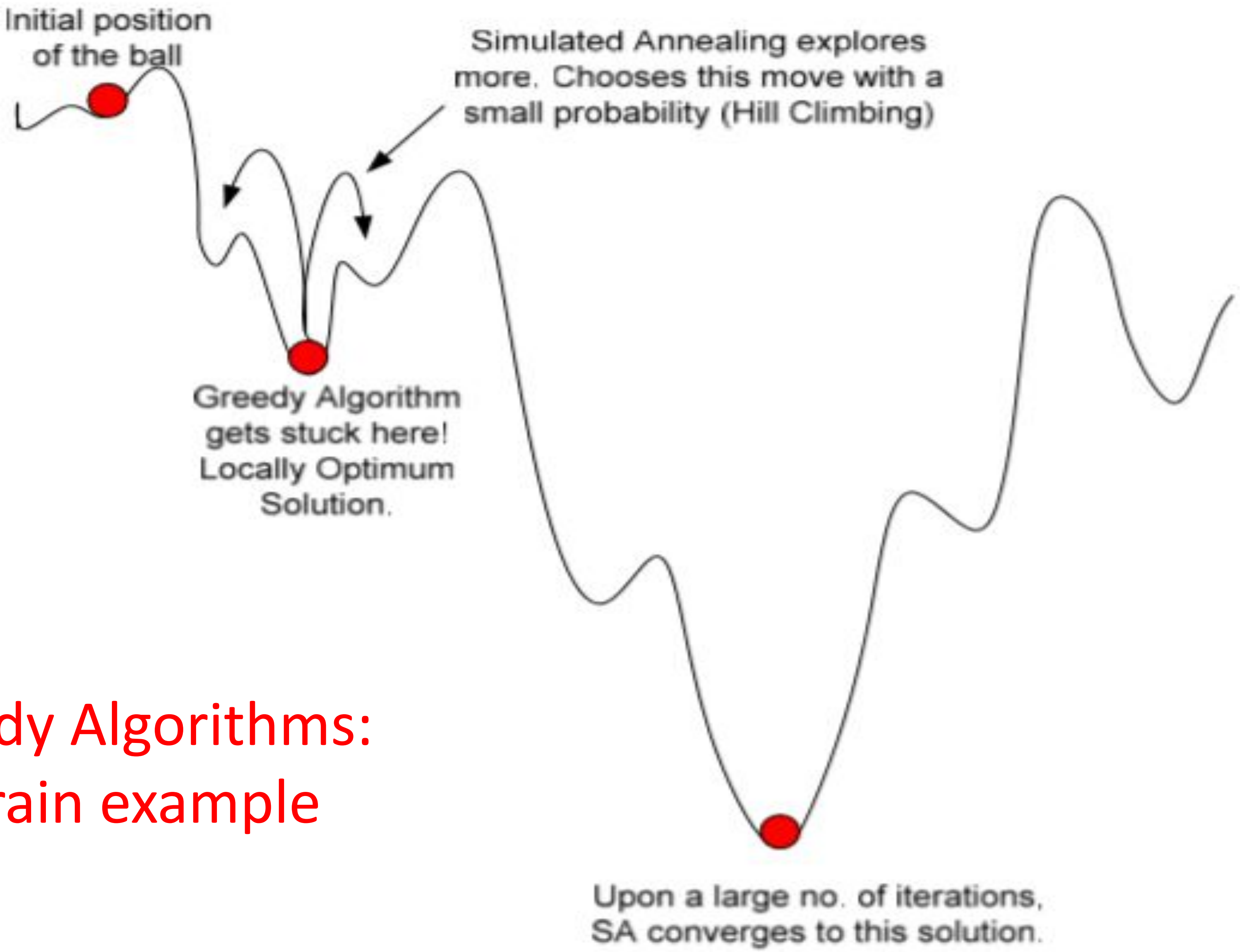
- The Nelder–Mead algorithm, a commonly applied numerical method used to find the minimum or maximum of an objective function in a multidimensional space, was employed to individuate new candidates for a solution, by probabilistically conditioning the acceptance of each movement of the simplex
- For a problem searching for minimum objective value, during the annealing process, each new solution x_j was accepted with a temperature-dependent probability P_T given by

$$P_T = \begin{cases} 1 & \text{if } f(x_j) \leq f(x_i) \\ e^{\left(\frac{f(x_i)-f(x_j)}{kT}\right)} & \text{if } f(x_j) \geq f(x_i) \end{cases}$$

- where T is the current temperature, k is the Boltzmann constant, and $f(x_i)$ and $f(x_j)$ are the fitness scores of the worst vertex x_i and new vertex of the simplex, respectively. The annealing temperature varied within a range $[T_{initial}, T_{final}]$, following a predefined cooling rate c and a predefined cooling scheme.
- Note that when $f(x_j)$ is much larger than $f(x_i)$, P_T is approaching 0.

- At the beginning of the process, which started from a randomly generated simplex, the higher temperature and higher probability of acceptance of new solutions allowed the simplex to explore a wide region of the search space, thereby escaping from local minima; however, as the temperature was reduced, the probability of acceptance of unfavorable solutions was reduced. The simulated cooling process followed a geometric temperature reduction scheme, activated after a maximum number of simplex walks $U_{\max}$ or a maximum number of conditioned acceptances $V_{\max}$. The algorithm parameters T_{initial}, T_{final}, c, $U_{\max}$ and $V_{\max}$ were defined as 2000 °C, 0 °C, 0.95, 20 and 5, respectively. Two stopping criteria, based on the maximum allowed number of simplex walks (250) and the minimum allowed difference between the best and worst fitness scores (100 s), were adopted.


?
barrier to local search
starting point
descend direction
local minima
global minima

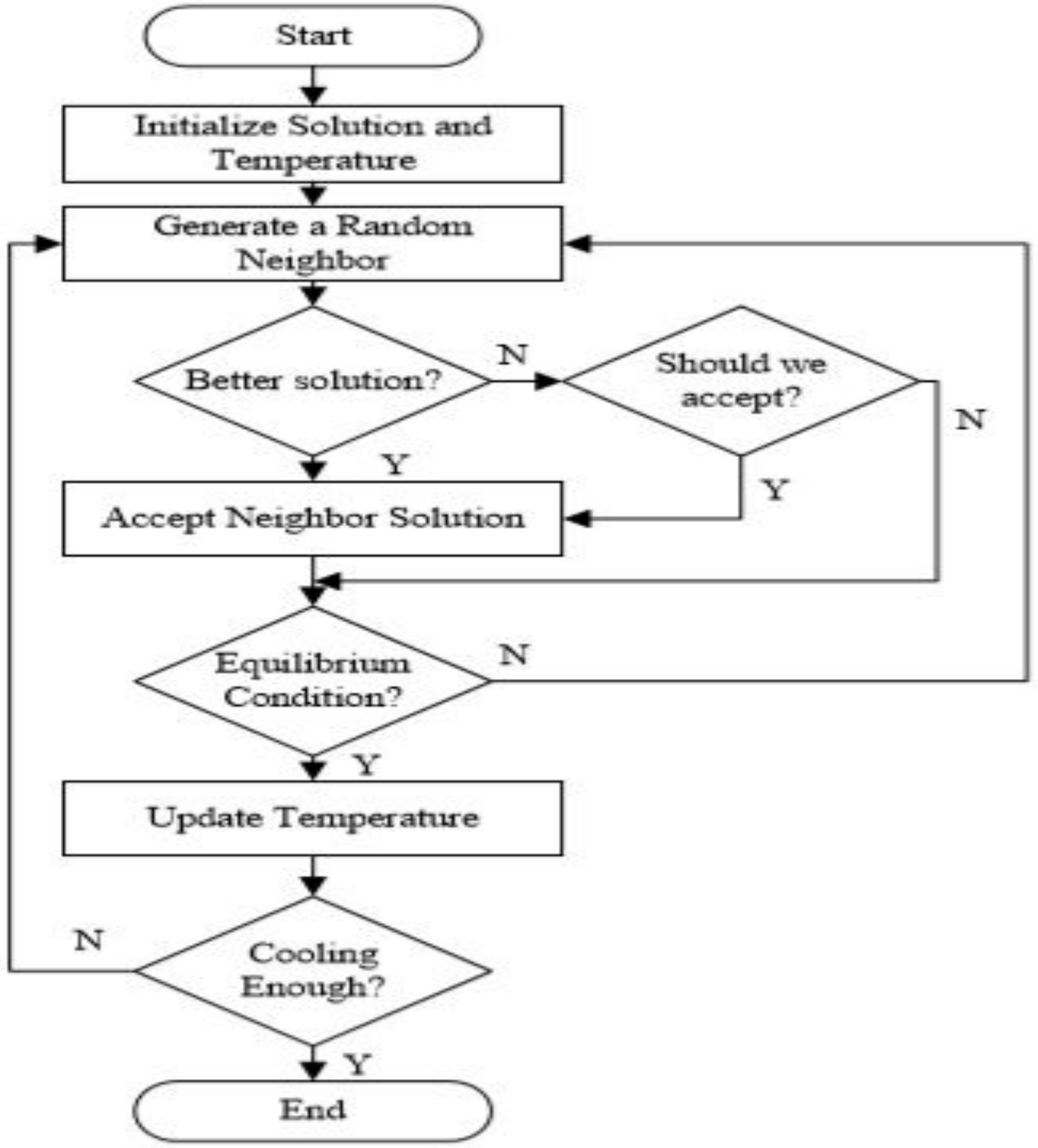

Initial position of the ball
Simulated Annealing explores more. Chooses this move with a small probability (Hill Climbing)
Greedy Algorithm gets stuck here! Locally Optimum Solution.
Upon a large no. of iterations, SA converges to this solution.

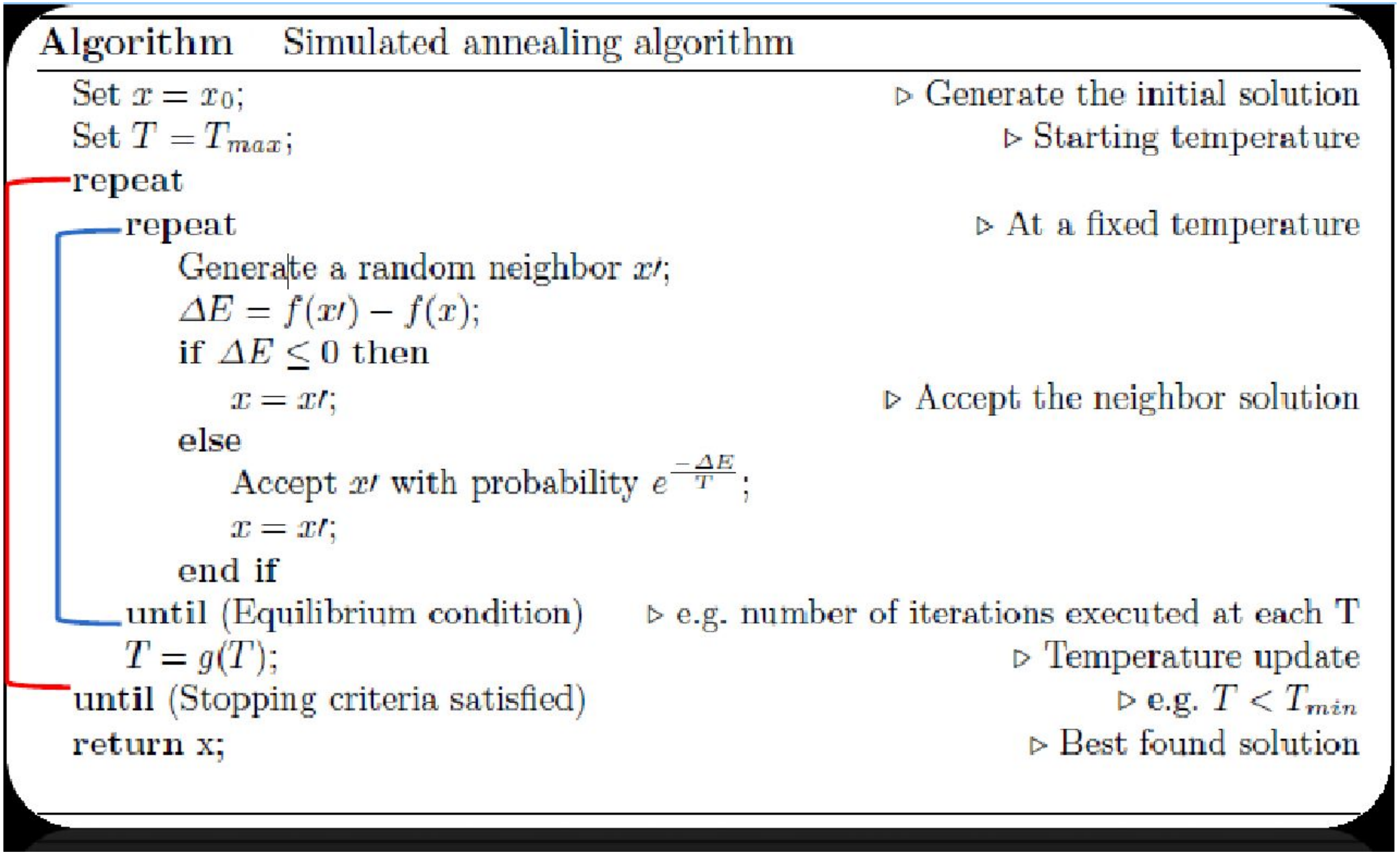
SA vs Greedy Algorithms: Ball on terrain example

- SA proceeds in several iterations from an initial solution x_0.
- At each iteration, a random neighbor solution x_j is generated.
- The neighbor solution that improves the cost function is always accepted as the current solution x_i .
- Otherwise, the neighbor solution is selected with a given probability that depends on the current temperature T and the amount of degradation E of the objective function.
- $E= f(x_i) - f(x_j)$ represents the difference in the objective value between the current solution x_i and the generated neighboring solution x_j

The analogy between the physical system and the optimization problem

Physical System	Optimization Problem
System state	Solution
Molecular positions	Decision variables
Energy	Objective function
Minimizing energy	Minimizing cost
Ground state	Global optimal solution
Meta stable state	Local optimum
Quenching	Local search
Temperature	Control parameter T
Real annealing	Simulated annealing


Start
Initialize Solution and Temperature
Generate a Random Neighbor
Better solution?
N
Should we accept?
N
Y
Accept Neighbor Solution
Y
Equilibrium Condition?
N
Y
Update Temperature
Cooling Enough?
N
Y
End



Algorithm Simulated annealing algorithm

```
Set $x = x_0$;                                  ▷ Generate the initial solution
Set $T = T_{max}$;                              ▷ Starting temperature
repeat
    repeat                                      ▷ At a fixed temperature
        Generate a random neighbor $x\prime$;
        $\Delta E = f(x\prime) - f(x)$;
        if $\Delta E \leq 0$ then
            $x = x\prime$;                      ▷ Accept the neighbor solution
        else
            Accept $x\prime$ with probability $e^{\frac{-\Delta E}{T}}$;
            $x = x\prime$;
        end if
    until (Equilibrium condition)   ▷ e.g. number of iterations executed at each T
    $T = g(T)$;                                 ▷ Temperature update
until (Stopping criteria satisfied)             ▷ e.g. $T < T_{min}$
return x;                                       ▷ Best found solution
```

An Example Problem

- Let us maximize the continuous function $f(x) = x^3 - 60x^2 + 900x + 100$.
- A solution x is represented as a string of 5 bits.
- The neighborhood consists in flipping randomly a bit.
- The initial solution is 10011 ($x = 19$, $f(x) = 2399$)
- Testing two sceneries:
  - First scenario: initial temperature T_0 equal to 500.
  - Second scenario: initial temperature T_0 equal to 100.
- Cooling rate 10% : $T_{new} = 0.9 \times T_{old}$

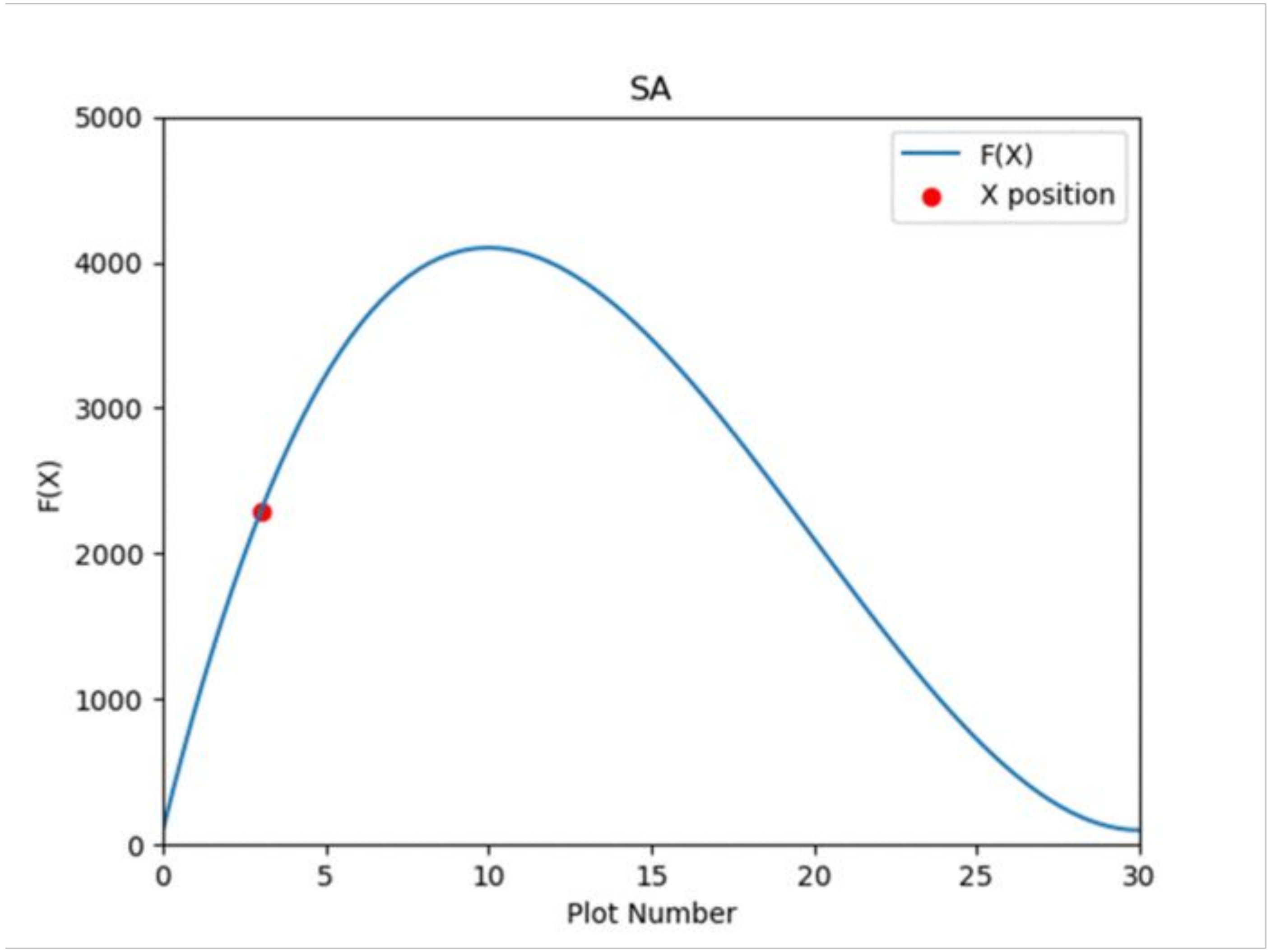
- In addition to the current solution, the best solution found since the beginning of the search is stored.
- Few parameters control the progress of the search, which are:
  - The temperature
  - The number of iterations performed at each temperature

First Scenario T = 500 and Initial Solution (10011)

T	Move	Solution	f	Δf	Move?	New Neighbor Solution
500	1	00011	2287	112	Yes	00011
450	3	00111	3803	<0	Yes	00111
405	5	00110	3556	247	Yes	00110
364.5	2	01110	3684	<0	Yes	01110
328	4	01100	3998	<0	Yes	01100
295.2	3	01000	3972	16	Yes	01000
265.7	4	01010	**4100**	<0	Yes	01010
239.1	5	01011	4071	29	Yes	01011
215.2	1	11011	343	3728	No	01011

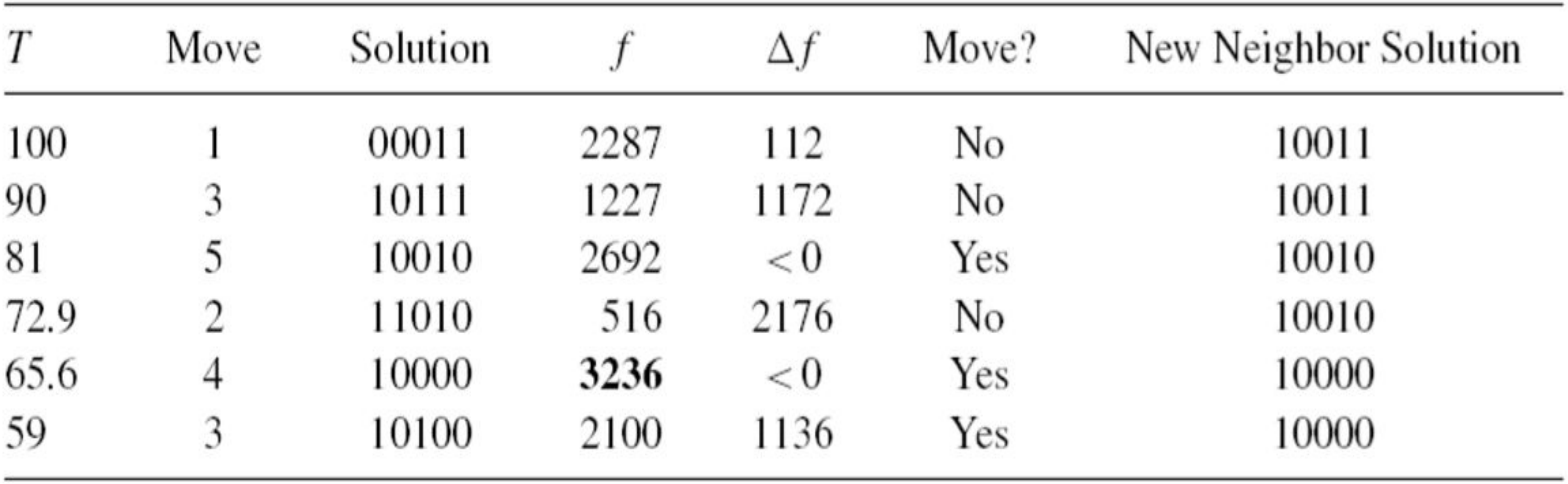
Hit a random # (0.0-1.0), each 1/5 flips corresponding bit 1 to bit 5

Cooling rate 10% : $T_{new} = 0.9 \times T_{old}$



T	X	f
500	3	2287
450	7	3803
405	6	3556
364.5	14	3684
328	12	3998
295.2	8	3972
265.7	10	4100
239.1	11	4071
215.2	27	343
100	3	2287
90	23	1227
81	18	2692
72.9	26	516
65.6	16	3236
59	14	2100

Second Scenario: T = 100 and Initial Solution (10011)



T	Move	Solution	f	Δf	Move?	New Neighbor Solution
100	1	00011	2287	112	No	10011
90	3	10111	1227	1172	No	10011
81	5	10010	2692	<0	Yes	10010
72.9	2	11010	516	2176	No	10010
65.6	4	10000	**3236**	<0	Yes	10000
59	3	10100	2100	1136	Yes	10000

When Temperature is not High Enough, Algorithm Gets Stuck

TABU Search

- Tabus are stored in a *short-term memory* of the search (the *tabu list*) and usually only a fixed and fairly limited quantity of information is recorded.
- Could record complete solutions
- The most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations

Notation

- $\boldsymbol{S}$, the current solution,
- $\boldsymbol{S^*}$, the best-known solution,
- $\boldsymbol{f^*}$, value of $\boldsymbol{S^*}$,
- $\boldsymbol{N(S)}$, the neighborhood of S,
- $\boldsymbol{\tilde{N}(S)}$, the "admissible" subset of $N(S)$ (i.e., non-tabu or allowed by aspiration).

Initialization

Choose (construct) an initial solution S_0.
Set $S := S_0, f^* := f(S_0), S^* := S_0, T := \emptyset$.

Search

```
While *termination criterion not satisfied* do
    ○ Select S in argmin $[f(S')]; S' \varepsilon \tilde{N}(S)$
    ○ if $f(S) < f^*$, then set $f^* := f(S), S^* := S$;
    ○ record tabu for the current move in $T$ (delete oldest entry if necessary);
endwhile.
```

Termination Criteria

- after some number of iterations without an improvement in the objective function value (the criterion used in most implementations);
- after a fixed number of iterations (or a fixed amount of CPU time);
- when the objective reaches a pre-specified threshold value.