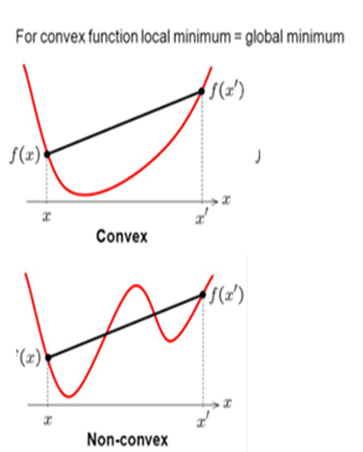
**Local search** differs from previous searches by focusing on the current state and exploring neighboring states to improve toward a solution incrementally. It's particularly useful for optimization problems where only the final configuration matters.

***Advantages***

1. *Minimal Memory Usage:* Unlike search trees, local search doesn't keep paths in memory, making it memory-efficient.
2. *Suitable for Large State Spaces:* Finds reasonable solutions in vast state spaces.
3. *Ideal for Optimization Problems:* Locates or approximates the best state based on an objective function, especially in convex spaces.

***Optimization***

Local search is often applied in optimization problems, aiming to find the best state by optimizing an objective function F(x), where x is a vector of continuous or discrete values.

Start with a complete configuration. Move from the current state to a successor state by changing a single element. Evaluates the quality of the state according to the objective fn.

***Examples***

1. *8 Queens Problem:* Arrange 8 queens on a chessboard so that no two queens attack each other. State: Queen's arrangement (one per column). Successors: Move one queen. Objective Function: Number of pairs attacking each other.
2. *Traveling Salesman Problem:* Visit each city exactly once, minimizing the total travel distance. State: Order of visited cities. Successors: Change to the current ordering. Objective Function: Length of the route.
3. *Cryptosystems:* Construct Boolean functions with desired cryptographic properties using local search, represented with truth tables. Successors: Change to the truth table. Objective Function: Assess relevant qualities of Boolean functions.
4. *Racing Yacht Hull Design:* Optimize yacht design components to minimize traversal time given wind conditions. State: Design components. Successors: Modification of design components. Objective: Estimate traversal time based on design parameters.

**Hill Climbing (Greedy Local Search)**

Generates nearby successor states to the current state, selects the best one, and replaces the current state with it. It terminates when a peak is reached but does not backtrack or look ahead beyond immediate neighbors.w

***Algorithm***

1. While (there exists an uphill point):
   1. Move to the neighbor with the highest evaluation function value.
   2. Let s\_next = max f(s) for all successors s of the current state n.
   3. If f(n) < f(s\_next) then move to s\_next.
   4. Otherwise, halt at the current state n.

***Properties***

1. *Termination:* Stops when a peak (local maximum) is reached.
2. *Local Search:* Only explores immediate neighbors of the current state.
3. *Random Choice:* Chooses randomly among the set of best successors.
4. *Greedy:* Selects best immediate neighbor without considering future consequences.

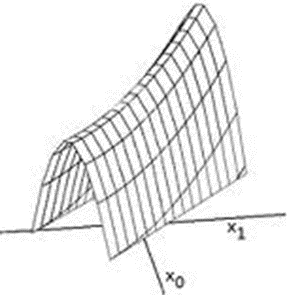
***Hill Climbing Variants***

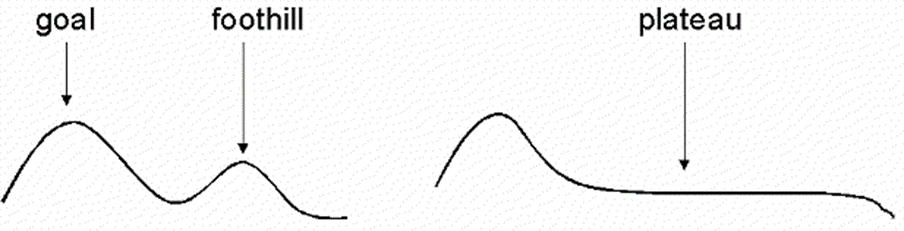
* 1. *Gradient Ascent/Descent:* Adjusts the current state by following the gradient of the function to maximize or minimize it.
  2. *Stochastic Hill Climbing:* Selects a random neighbor state and decides whether to move to it based on probability.
  3. *First Choice Hill Climbing:* Generates successors randomly until one is better than the current state, useful for states with many successors.
  4. *Random-Restart Hill Climbing:* Starts hill-climbing searches from random starting positions, saving the best result found so far.

***Drawbacks:*** May get stuck in local maxima, plateaus, or ridges.

***Remedy:*** Random-restart hill climbing conducts multiple hill-climbing searches from random starting positions, increasing the likelihood of finding the global optimum.

***Challenges in Hill Climbing:***

1. *Local Maximum:* A peak lower than the highest peak, leading to suboptimal solutions.
2. *Plateau:* Flat evaluation function area, causing random walks.
3. *Ridges:* Gentle slopes toward a peak, causing oscillations.



**Simulated Annealing (SA)**

Inspired by a metallurgical process where metals are slowly cooled to reach a low-energy crystalline state. This is mimicked in optimization to find global minima.

***Algorithm***

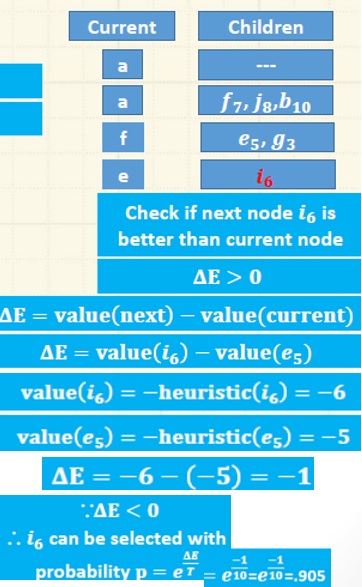
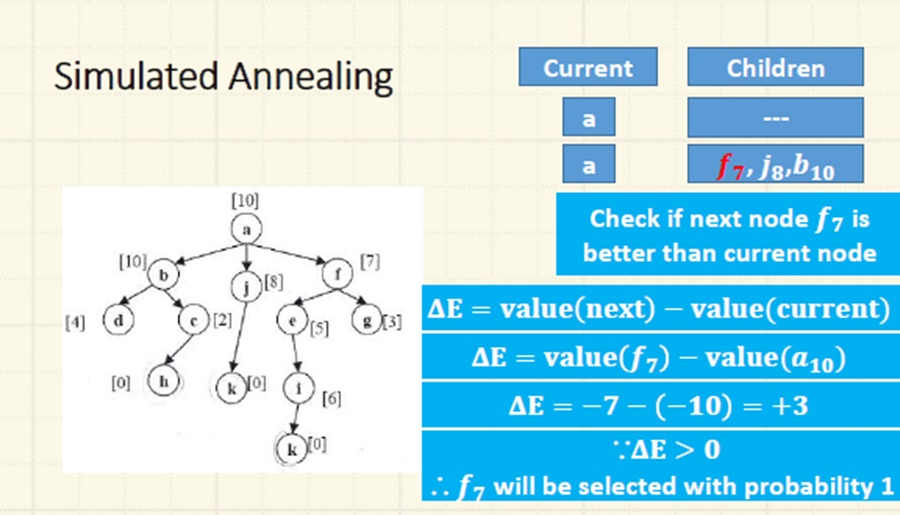
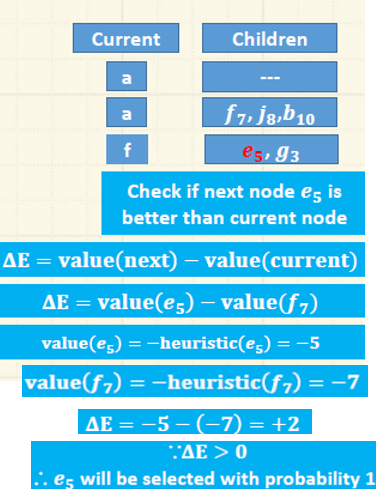
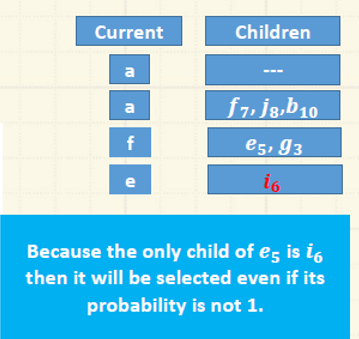
1. Initialization: Start with an initial state and high-temperature T.
2. Iteration:
   1. Generate a random neighbor state.
   2. If better, move to it.
   3. If worse, move with probability ΔE/T.
3. Cooling: Reduce temperature according to a schedule.

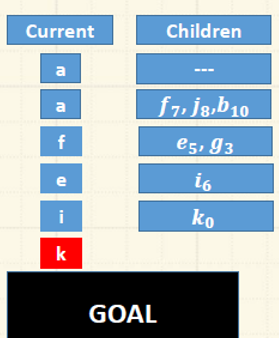
***Key Points***

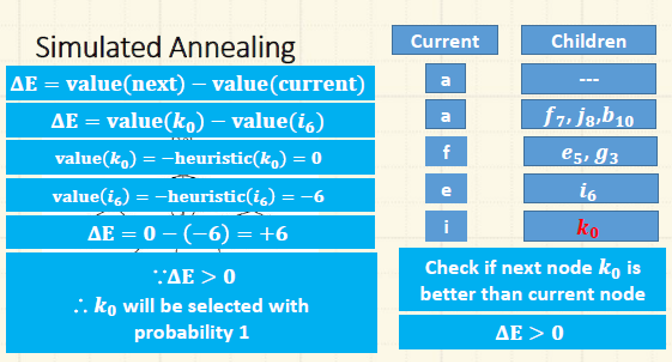
1. *Temperature (T):* Controls the probability of accepting worse states.
2. *Cooling Schedule:* Defines how temperature decreases.
3. *Stopping Conditions:*
   1. Temperature reaches zero.
   2. Threshold change in value.
   3. Fixed number of iterations.

***Advantages***

1. *Memory Efficient:* Stores only current state and neighbors.
2. *Effective for Large Spaces:* Finds solutions in large or complex spaces.
3. *Global Optimization:* Avoids local optima.





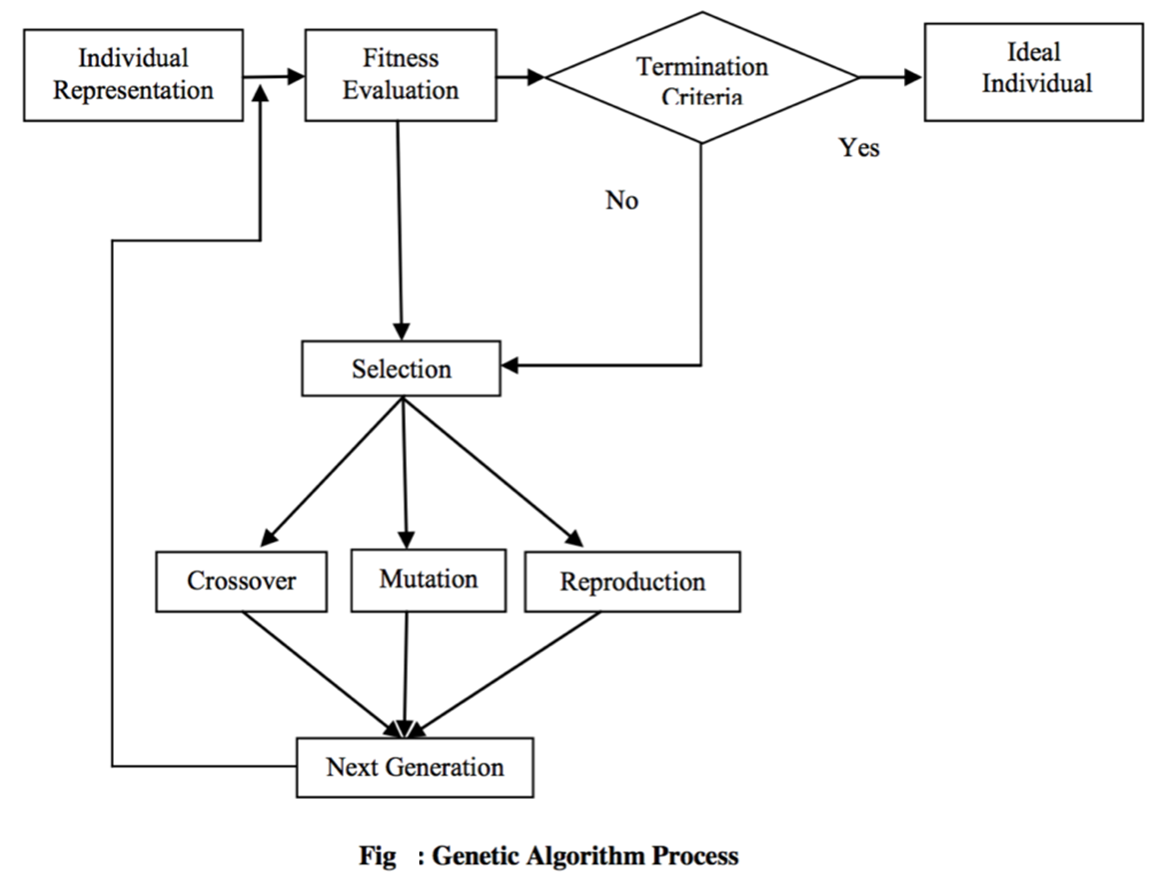
**Genetic Algorithms** are heuristic search and optimization techniques that simulate the process of natural evolution. They are based on the principle of "survival of the fittest" and are used to find true or approximate solutions to optimization and search problems.

***Process***

1. *Initialization:* Start with a random population of individuals.
2. *Selection:* Select individuals based on their fitness to reproduce.
3. *Crossover:* Combine pairs of individuals to produce offspring.
4. *Mutation:* Apply random changes to offspring.
5. *Replacement:* Form a new population from parents and offspring.
6. *Termination:* Repeat the process until a solution is found or a time limit is reached.

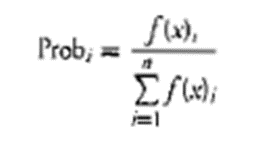
***Applications***

1. *Optimization Problems:* Suitable for complex optimization problems.
2. *Easy to Apply:* Simple to implement for various problems.
3. *Results Vary:* Can produce good results for some problems but not all.



Consider the problem of maximizing the function, f(x)= x2

where x is permitted to vary between 0 and 31.

GA approach: Random initialization, binary code, e.g. 01101 ↔ 13. Population size: 4

Roulette wheel selection. 1-point crossover, bitwise mutation

One generational cycle performed manually is shown here:-