Simulated Annealing

Simulated Annealing

Use a more complex Evaluation Function:

- Do sometimes accept candidates with higher cost to escape from local optimum
- Adapt the parameters of this Evaluation Function during execution
- Based upon the analogy with the simulation of the annealing of solids

Other Names

- Monte Carlo Annealing
- Statistical Cooling
- Probabilistic Hill Climbing
- Stochastic Relaxation
- Probabilistic Exchange Algorithm

Simulated Annealing

- Word comes from **metallurgy**
- In metallurgy – to make really strong objects, we follow slow and controlled cooling procedure known as annealing
- Same can be applied to Computer Science Algorithms
- SN algorithms are applied in
  - Travelling Sales Person problem
  - Designing printed circuit boards
  - Planning for a path for a Robot
  - Designing 3d structures of protein molecules in Bio-Informatics

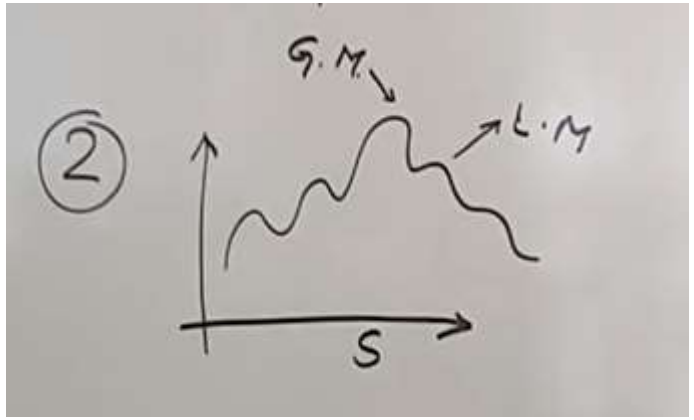
- **Intuition behind SN**: At the beginning, we don't care about actually moving towards the good solution and accept bad moves as well and bad configurations as well.
  - But as we progress towards the solution, we become more careful and try to get closer to the solution and selecting only good moves

Simulated Annealing

- To formulate any problem in order to apply algorithms like SN, every problem needs to have a way to define how good a given configuration is.
- Every problem or every solution that we are solving needs to have a way to describe the goodness or fitness of the solution of the problem.
- In other words, in optimization problems, it is known as **Energy** of the solution
- Define the ENERGY of the solution as E and CHANGE in Energy as ΔE
  - Every time the system goes through a change of configuration, we can compute Energy
- **Example**: **Water bubble Game**:
  - One can focus in just one of the hooks and continue to play the game that he/she can get as many rings as possible in one of the holes, forgetting about the remaining hooks
  - After certain time, We will end up in a **DEADLOCK** where we have lot of rings hooked in one of the hooks but the other hooks are empty
  - If we play vigorously, remove lot of rings that are already solved
  - If we play very slow, it will take really a long time to get to the final solution

Simulated Annealing

- Deadlock Situations are known as LOCAL MAXIMA or LOCAL MINIMA
- Can plot all possible configurations of a given system on X axis and on Y axis, we were to compute the corresponding ENERGY of all the configurations, then ENERGY LANDSCAPE will be



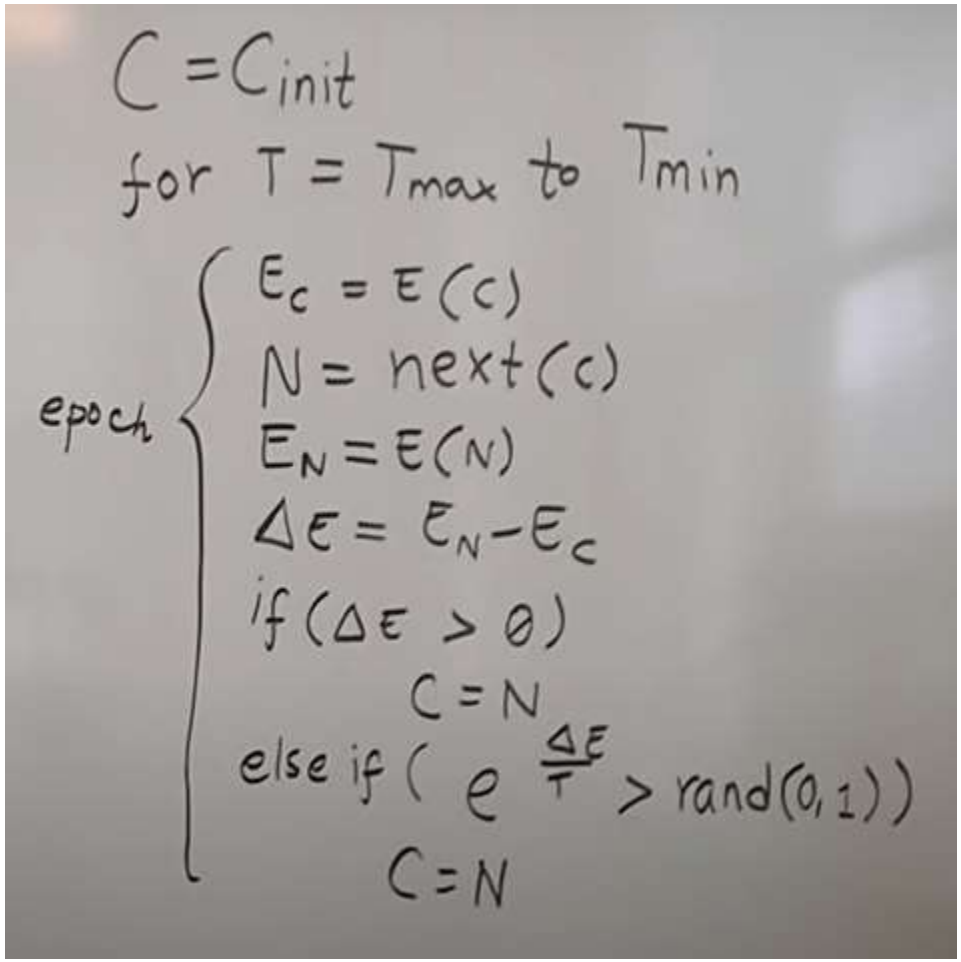
- Once Energy is defined and comes to know about the GLOBAL MINIMA / MAXIMA, we also need a way in which we can change the configuration of the system every time
- In Water bubble game, we can change the CONFIGURATION, by pressing the button – it **randomly** alters the configuration of the system to get a new configuration

Simulated Annealing

- Where we can apply this ?

- With these definitions, Let us get into **SIMULATED ANNEALING** Algorithm

Simulated Annealing Algorithm



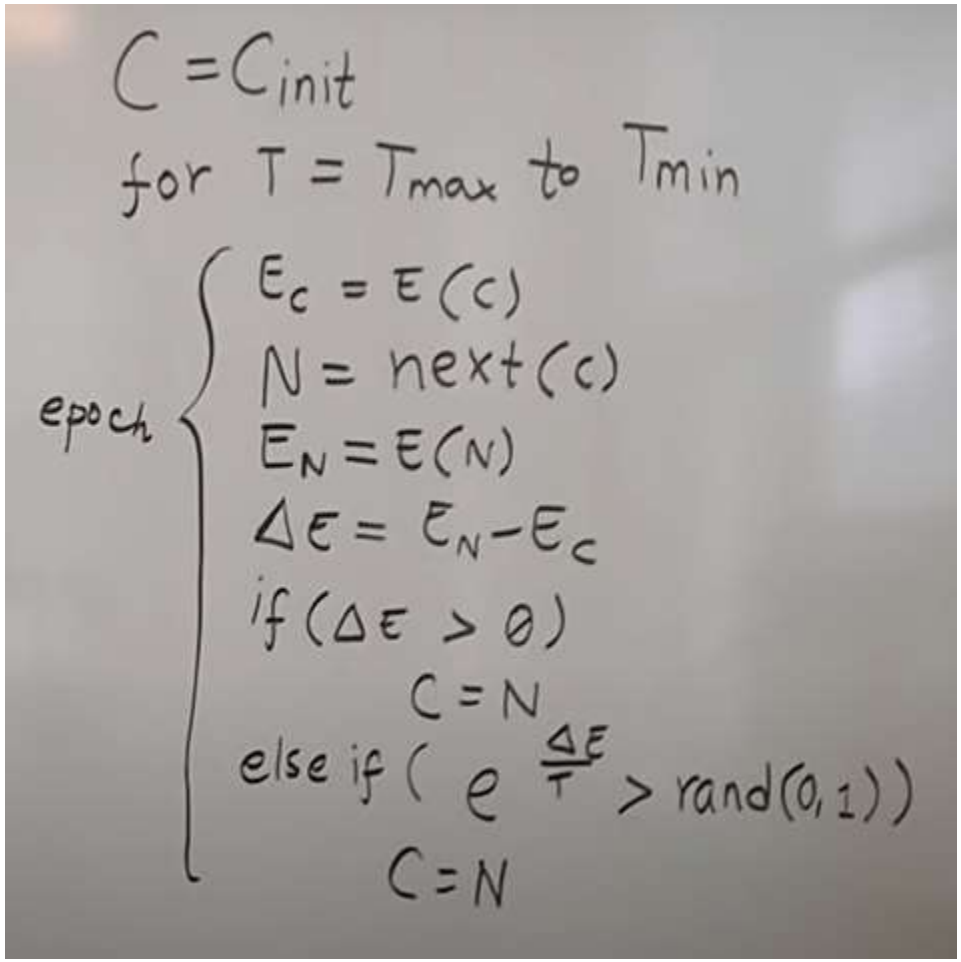
C_{init} – Initial Configuration

T – Temperature (Monotonically decreasing function)

In annealing, start with very high temperature, we slowly controlled to cool the material

- For every value of temperature, compute the current energy of the system for the current configuration (E_c)
- Alter/Change the configuration
- Compute Energy for this New config (E_N)
- Compute Change in Energy

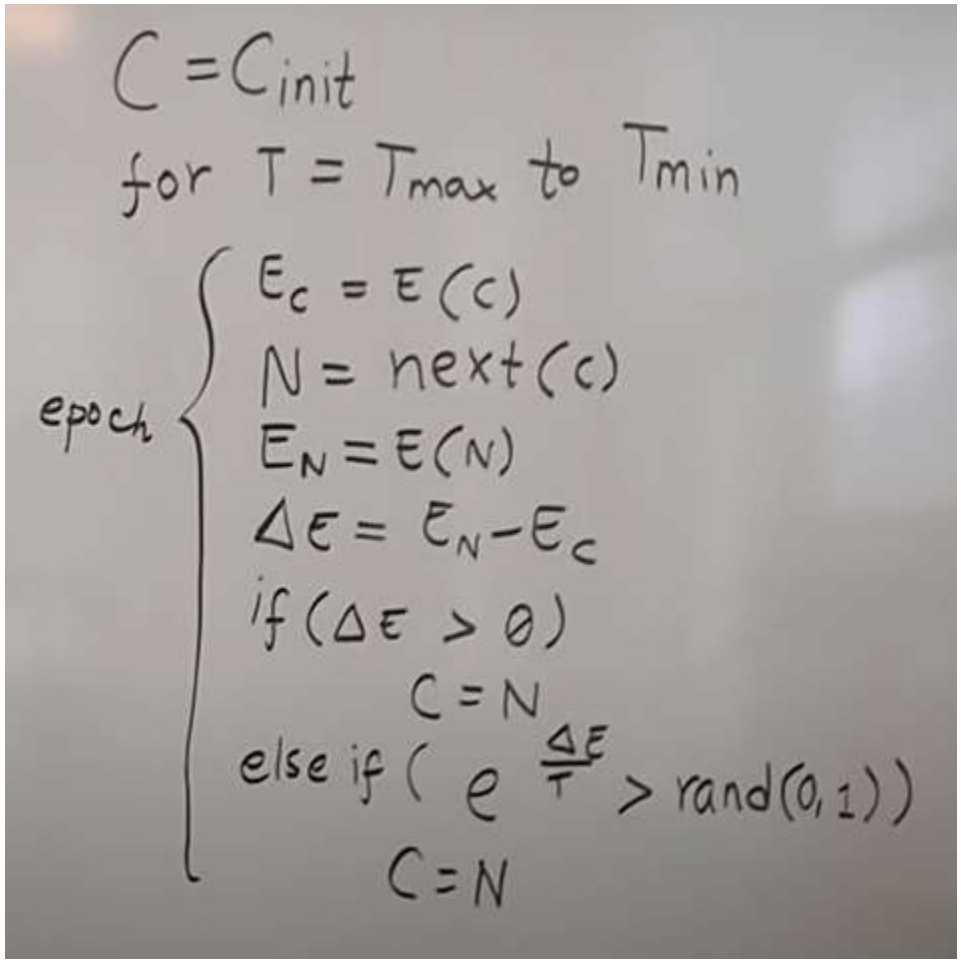
Simulated Annealing Algorithm



IF we are working on **MAXIMIZATION problem**

- **ΔE > 0**, Change in Energy is a good move. So accepts the move
  - Set the NEW config as the CURRENT config
  - Repeat the Process and getting a better solution and move close to the FINAL solution
- **ΔE < 0**, (Negative) Change in Energy is a bad move. So compute the probability as shown
  - If probability is very high, then accept the move even if it is a bad move
  - If probability is very low, then accept the move even with low probability

Simulated Annealing Algorithm



Probability depends on 2 terms, Δ**E and T**

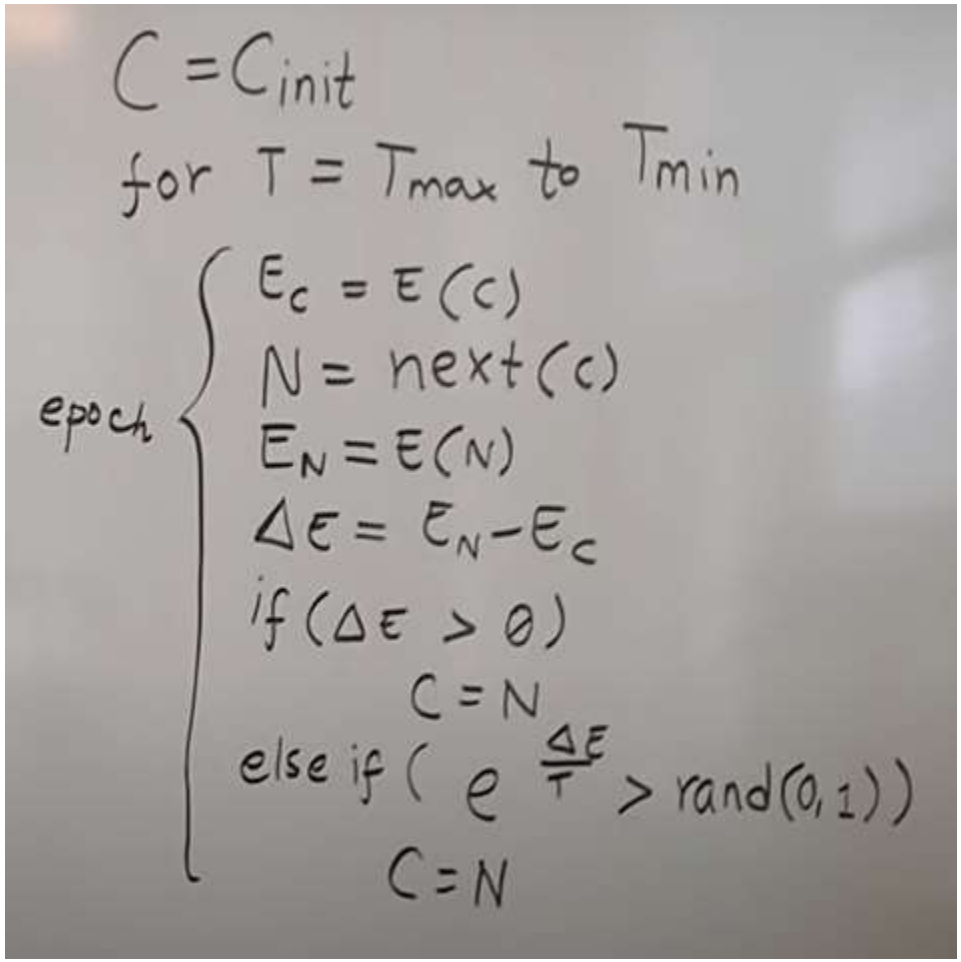
- When T is very high, then probability for accepting the bad move becomes very high
  - In other words, at HIGH T, we are exploring the solution space or we are exploring the configurations and accepting the bad moves as well
- When T is very low, then we have very low probability to accept the bad moves

- Eg. T = 10,000 and ΔE = 10, then

  $e^{0.001}$ = 1.0010005001667

Means, has a very high probability to become greater than rand(0,1) . So accepts the move

Simulated Annealing Algorithm



Probability depends on 2 terms, Δ**E and T**

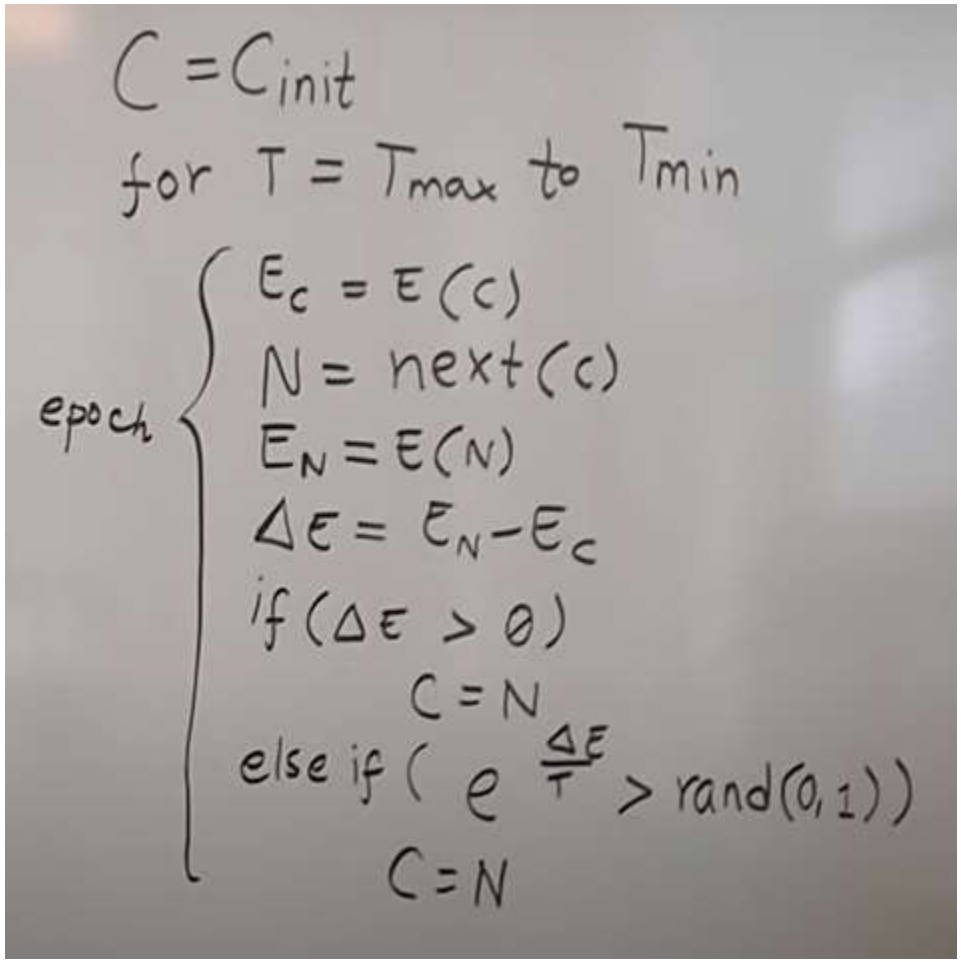
- **When T is very high,** then probability for **accepting the bad move** becomes **very high**
  - In other words, at HIGH T, we are exploring the solution space or we are exploring the configurations and accepting the bad moves as well

- Eg. T = 10,000 and ΔE = 10, then

  $e^{0.001}$ = 1.0010005001667

Means, has a very high probability to become greater than rand(0,1) . So accepts the move

Simulated Annealing Algorithm

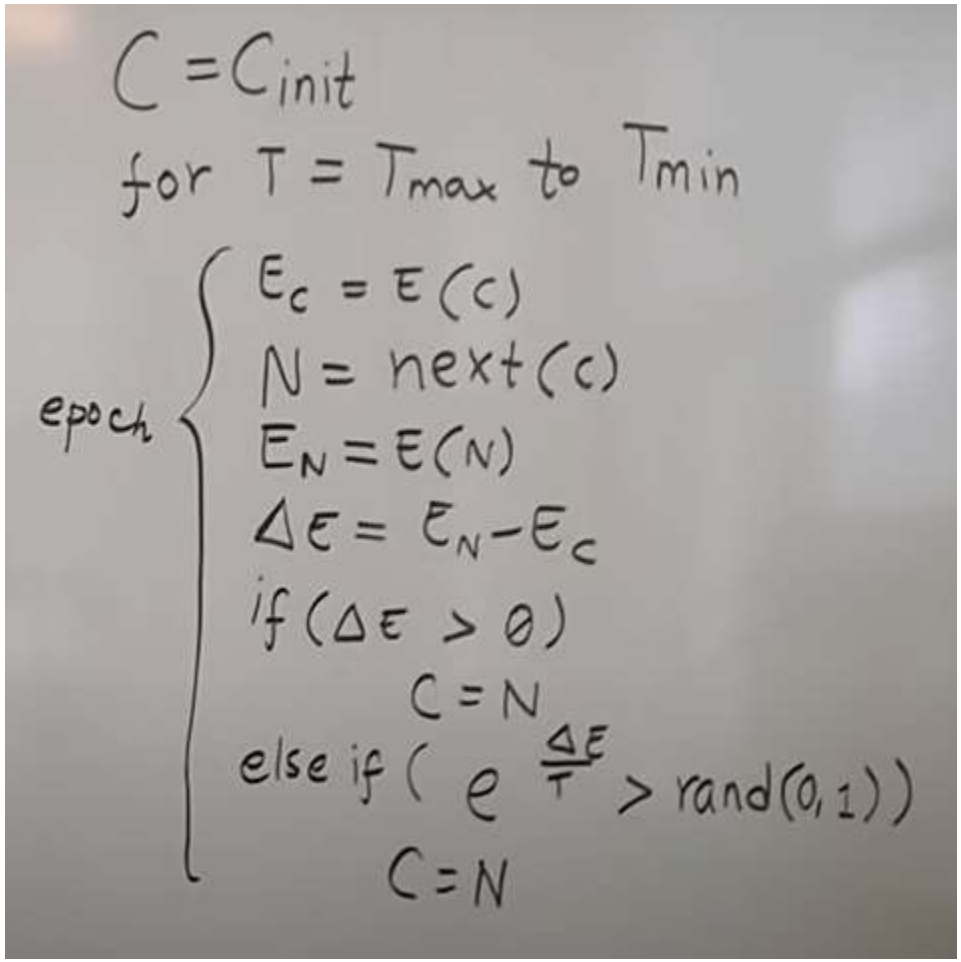


ΔE also influences

- **When ΔE is very high,** thenwe have very low probability to accept **the bad move**
- When **ΔE** is very low, then we have very high probability to accept the bad move

- Repeat the process for certain no. of times. Epochs = 100 or 200
- Finally, expecting the solution to converge towards the GLOBAL MAXIMUM (/ MINIMUM)

Simulated Annealing Algorithm



1. **Remove the Probability factor or the Temperature factor and always accept the good moves only – HILL CLIMBING / Greedy algorithm** which always move towards a better solution
   - Such algorithms are prone to easily be stuck in LOCAL MINIMA
2. **Random Walk –** We don't care about how good a move we are making every time, but we just explore, continue to explore the space.
   - Such algorithms never converge and will probably never give you the best optimal solution

Simulated annealing search

- Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency

```
function SIMULATED-ANNEALING( problem, schedule) returns a solution state
   inputs: problem, a problem
           schedule, a mapping from time to "temperature"
   local variables: current, a node
                    next, a node
                    T, a "temperature" controlling prob. of downward steps

   current ← MAKE-NODE(INITIAL-STATE[problem])
   for t ← 1 to ∞ do
       T ← schedule[t]
       if T = 0 then return current
       next ← a randomly selected successor of current
       ΔE ← VALUE[next] – VALUE[current]
       if ΔE > 0 then current ← next
       else current ← next only with probability e^(Δ E/T)
```

Properties of simulated annealing search

- One can prove: If T decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1
- Widely used in VLSI layout, airline scheduling, etc

Simulation of cooling (Metropolis 1953)

Minimization Problem

- At a fixed temperature T :
- Perturb (randomly) the current state to a new state
- ΔE is the difference in energy between current and new state
- If $\Delta E < 0$ (new state is lower), accept new state as current state
- If $\Delta E \geq 0$, accept new state with probability

$$Pr\ (accepted) = exp\ (-\ \Delta E\ /\ k_B . T)$$

- Eventually the systems evolves into thermal equilibrium at temperature T ; then the formula mentioned before holds
- When equilibrium is reached, temperature T can be lowered and the process can be repeated

Performance

- SA is a general solution method that is easily applicable to a large number of problems
- "Tuning" of the parameters (initial *c*, decrement of *c*, stop criterion) is relatively easy
- Generally the quality of the results of SA is good, although it can take a lot of time
- Results are generally not reproducible: another run can give a different result
- SA can leave an optimal solution and not find it again (so try to remember the best solution found so far)
- Proven to find the optimum under certain conditions; one of these conditions is that you must run forever

Conclusions

- It is very easy to implement.
- It can be generally applied to a wide range of problems.
- SA can provide high quality solutions to many problems.
- Care is needed to devise an appropriate neighborhood structure and cooling scheduler to obtain an efficient algorithm.