Simulated Annealing

Winter 2024

Population versus Individual

Hill Climb

- The simplest individual-based search method
- Start at a random solution; consider various moves within the neighborhood; accept the ones when advantageous; repeat until no more move can be made

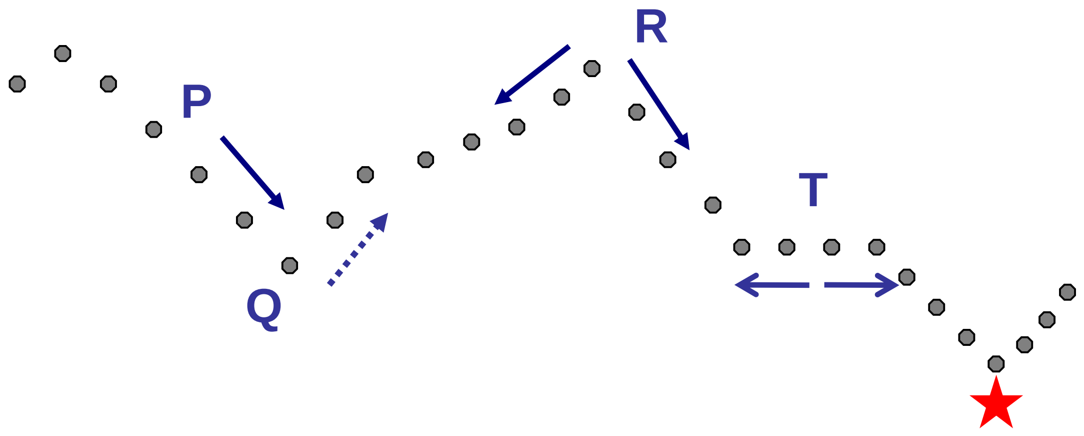
Pseudocode of Hill Climbing

```
Select a starting solution $s_0 \in S$
While termination condition is not met:
    iteration limit or $f(s_0) < f(s)$ for all $s \in N(s_0)$
        Select $s$ such that $f(s) < f(s_0)$
        Replace $s_0$ by $s$
Return $s_0$ as the approximation to the optimal solution
```

*** For minimization problem*

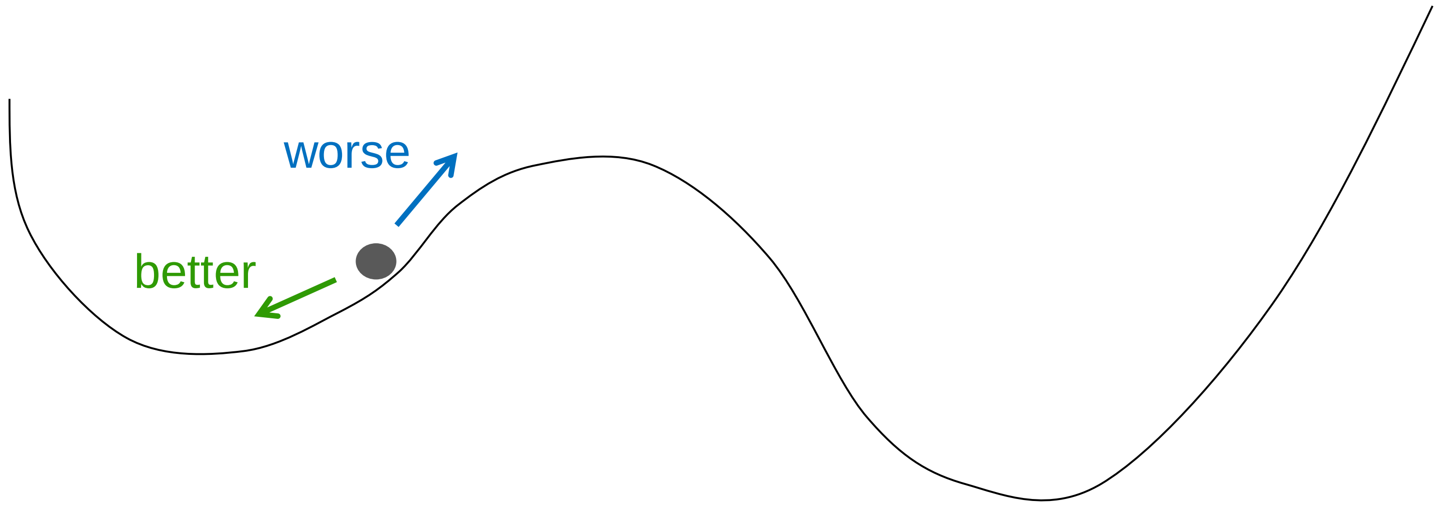
Disadvantages of hill climbing

- It is very likely to get stuck in local optimal solutions
- Solution quality is highly sensitive to the initial solution (P always leads to Q; R can be improved a lot more to ★)
- Hard to decide where to go if all the neighbors have the same objective value (After T, where to go?)



Possible Improvements

- Hill climbing in inner-loop and re-start the initialization in outer-loop
- Increase the range (vision) of neighborhood
- **Allow "bad move"** whose frequency is governed by a probability function



Simulated Annealing: Thermodynamics

- Metropolis's work in statistical thermodynamics
- At temperature t, the probability of an increase in energy of magnitude $\triangle E$ is given by

$$P(\triangle E) = \exp(-\triangle E / \alpha t)$$

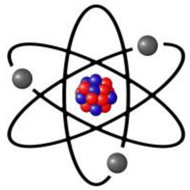
where α is Boltzmann's constant

1.38065×10^{-23} joules/kelvin

Dr. Nicolas Metropolis
1915-1999

Dr. Ludwig Boltzmann
1844-1902

Simulated Annealing



- If solid material is heated past its melting point and then cooled back into a solid state, the structural property depends on the **cooling rate**
- The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy
- **Slow cooling** gives them more chances of finding configurations with lower internal energy than the initial one

Physical Analogy

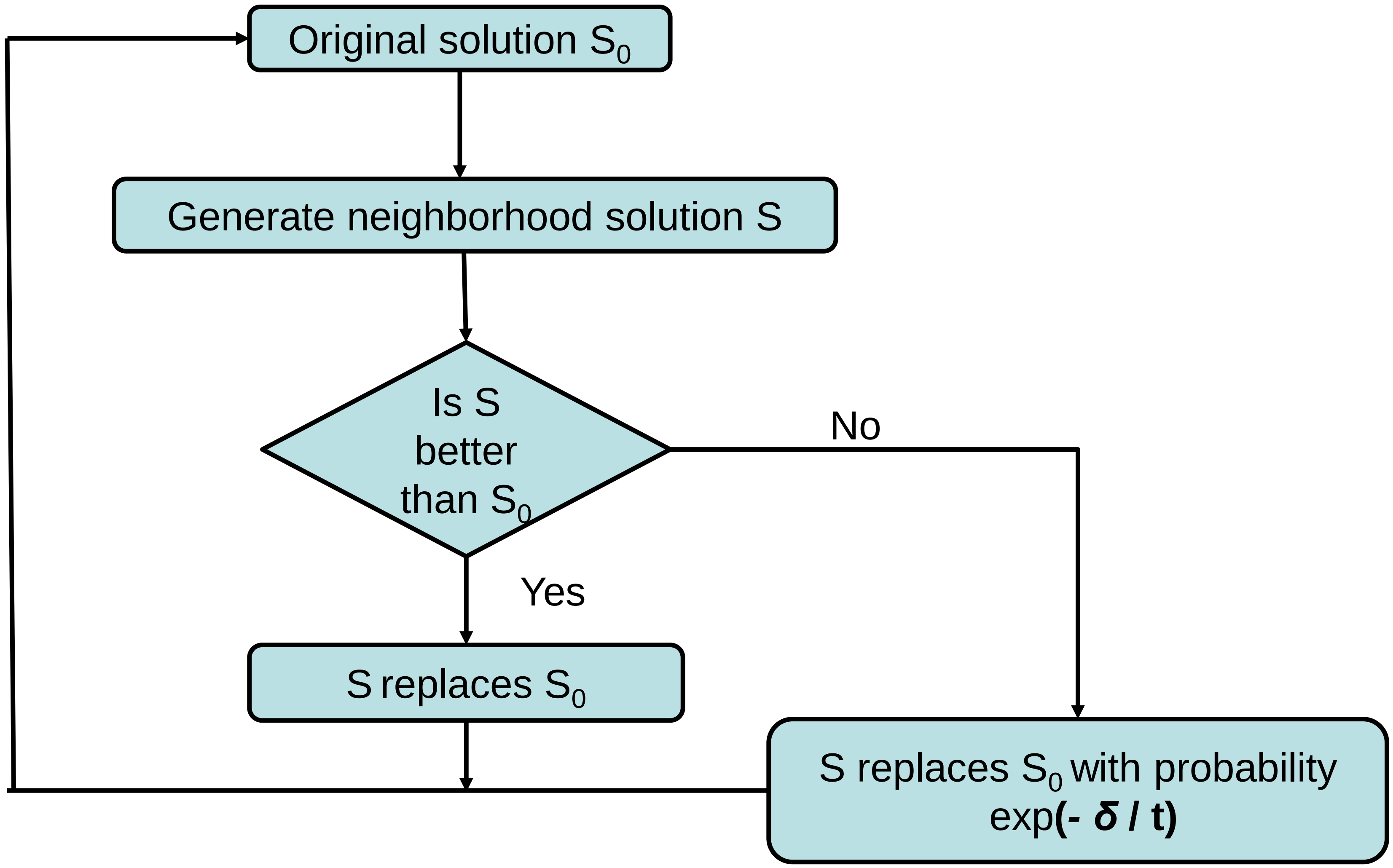
- **If the cooling is done too fast**, the resulting solid will be frozen into **a locally optimal structure**, such as a glass or a crystal with several defects in the structure.
- Inspiration: Finding low energy states of a solid by initially melting the substance, and then lowing the temperature **slowly**

Concept

Prof. Kirkpatrick
Computer Science

- Kirkpatrick, Gelatt, Jr., and Vecchi, (1983) "Optimization by Simulated Annealing", *Science*, 220(4598), 671-680.
- Each step of the Simulated Annealing (SA) algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends on **the difference between the corresponding function values** and **on a global parameter** *T* (*temperature*), that is **gradually decreased during the process**.

Flowchart


Original solution S_0
Generate neighborhood solution S
Is S better than S_0
No
Yes
S replaces S_0
S replaces S_0 with probability exp**(- *δ* / t)**

Pseudocode of SA (1)

```
Select an initial solution $s_0$; initial temperature $t_0$
Select a temperature reduction function (cooling
  schedule) T($k$): $k$ is the reduction step
Outer: While temperature > final_temperature
  Inner: While iteration limit $n_{iter}$ is not reached
      Select $s \in N(s_0)$
      $\delta = f(s) - f(s_0)$
      If $\delta < 0$ (for minimization problem)   $\delta > 0$ for maximization
        $s_0 = s$   #always accept better moves
```

Pseudocode of SA (2)

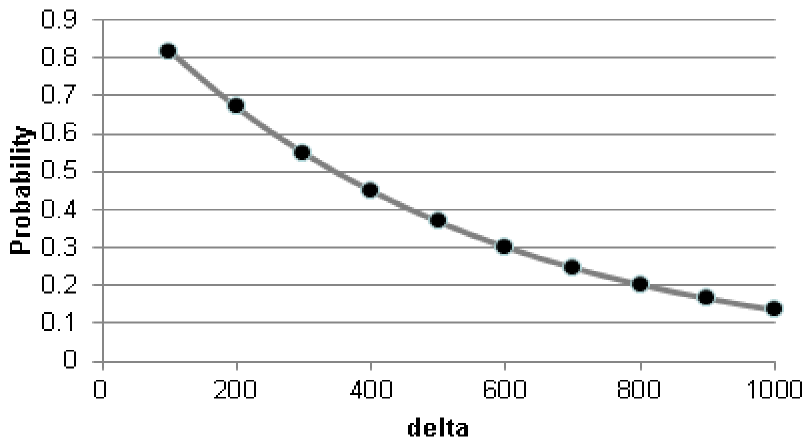
```
            Else    #may accept worse moves
                generate $r \sim U(0,1)$
                If $r <$ **$\exp(-\delta / t)$**, $s_0 = s$ *(for minimization)*
                If $r <$ **$\exp(\delta / t)$**, $s_0 = s$ *(for maximization)*
    End Inner
    $t = T(k)$
End Outer
Return $s_0$
```

Probability of Adverse Move

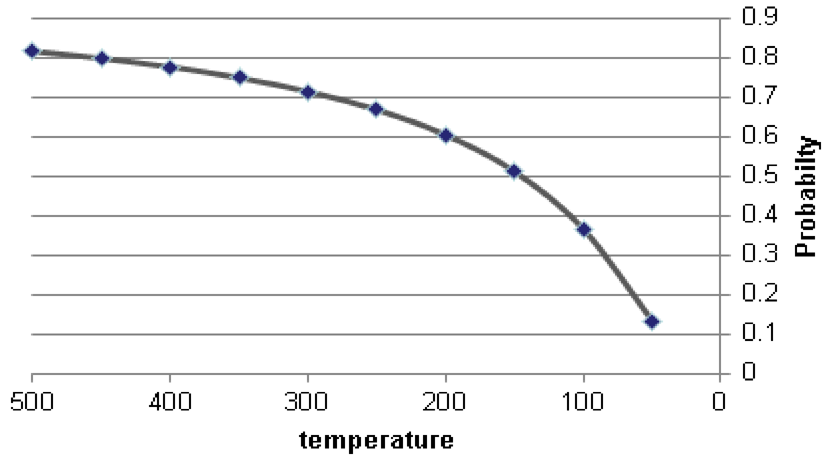
- Probability $\exp(-\delta / t)$ is related to δ and t

  larger $\delta \longrightarrow$ smaller acceptance probability
  (less likely to accept worse move)

  lower temperature $\longrightarrow$ smaller acceptance probability
  (decreasing probability as temperature drops)



t = 500



delta = 100

Key issues

For worse moves, if $r < \exp(-\delta / t)$, $s_0 = s$

- Decreasing temperature → Decrease the probability of accepting bad moves
- Allow a bad move but the probability of accepting it is inverse proportional to the difference from the previous move
- Increasing the number of iterations at each temperature → spend more efforts in later iterations

Numerical Example

Maximize $f(x)=x^3-60x^2+900x+100$; $x=[0\sim31]$

Starting temperature $t_0=500$

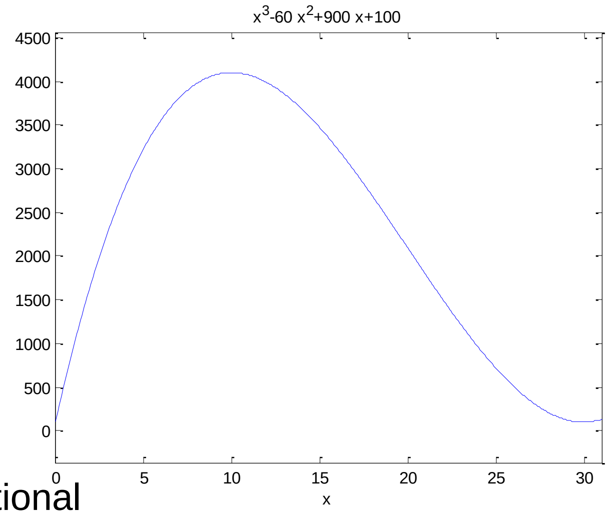
Cooling schedule $T(k)=t_0 0.9^k$; $n_{iter}=1$ (1 iteration at each temperature)

x is expressed as 5-bit representation; its **neighborhood is selected based on one-bit change**

Initial random solution: (10011)

Initial x : $2^4+2^1+2^0=19$

Initial $f(x)$: 2399



Experiment Result

If $r < \exp(\delta / t)$, $s_0 = s$

exp(-112/500)=0.79932

exp(-247/405)=0.54342

T(*k*)	Selected bit	String	*x*	*f*(*x*)	δ	Move Prob.	*r* *random*	Move?
500	1	00011	3	2287	-112	0.79932	0.638	YES
450	3	00111	7	3803	>0			
405	5	00110	6	3556	-247	0.54342	0.124	YES
364.5	2	01110	14	3684	>0			
328.0	4	01100	12	3988	>0			
295.2	3	01000	8	3972	-16	0.94724	0.837	YES
265.7	4	01010	10	4100	>0			
239.1	5	01011	11	4071	-29	0.88578	0.419	YES
215.2	1	11011	27	343	-3728	3.00E-08	0.573	NO
193.7	2	00011	3	2287	-1784	0.0001	0.249	NO

Optimal

Move away

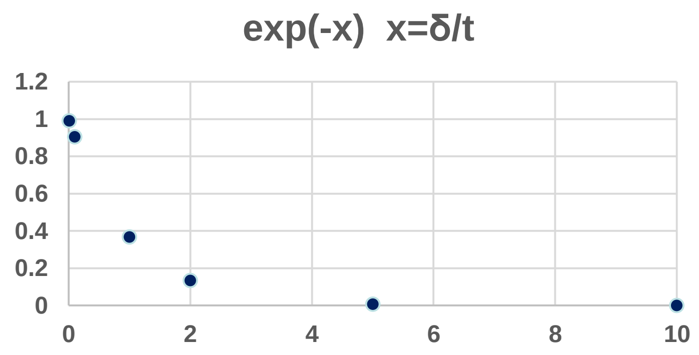
Generic Decisions

- Initial temperature
- Final temperature
- Cooling schedule
  - **Deceasing** function
- Number of iteration at each temperature: n_{iter}
  - Spend longer time at lower temperatures
  - **Increasing** function

Initial Temperature

For worse moves, if $r < \exp(-\delta / t)$, $s_0 = s$

- If too hot, we are conducting a pure random search (always accept worse moves)
- If too cold, we are implementing hill climbing (do not allow worse moves at all)
- Preliminary run: Start high, freeze quickly until about 50%~80% of worse moves are accepted; use this as the initial temperature



Final Temperature

- It is usual to let the temperature decrease until it is close to zero
- Sufficient to stop when the chance of accepting a bad move is negligible

Cooling Schedule (1)

- Geometrical
  $T(k) = t_0 a^k$; a is a constant smaller than (close to) 1
- Logarithmic
  $T(k) = t_0 \ln(2)/\ln(k)$ $k \geqq 2$
- Exponential

  $T(k) = t_0 \exp(-c/k)$; c is a small positive constant close to 0

Cooling Schedule (2)

- Lundy and Mees (1986)

  $T(k) = T(k\text{-}1) / (1+ \gamma\, T(k\text{-}1))$; γ is a small constant

- Aarts and Korst (1989)

  $T(k) = T(k\text{-}1) / [1+(T(k\text{-}1)\ln(1+ \gamma)/3\sigma_t)]$;
  σ_t is the standard deviation at temperature *t;*
  γ is a constant

Number of Iterations (1)

- Allow enough iterations at each temperature so that the system stabilises at that temperature
- The number of iterations may vary at different temperatures

Number of Iterations (2)

- A fixed constant
- Linearly increasing
  $n_{iter} = \alpha(t_0 - t) + \beta$; α and β are constants
- Other increasing functions
- When a certain number of better solutions have been found, stop the iterations; go to next temperature

Number of Iterations (3)

- Compromise:
  - either doing a large number of iterations at a few temperatures
  - or a small number of iterations at many temperatures
  - a balance between the two

Variant: Acceptance Probability

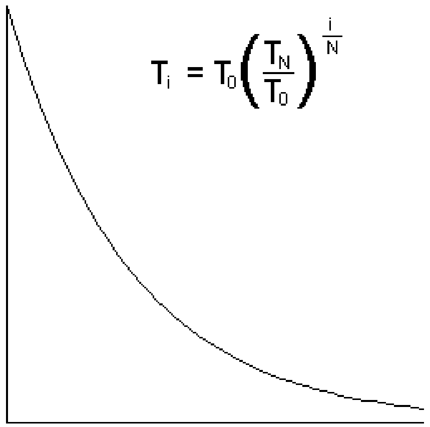
- Simplification on the distribution

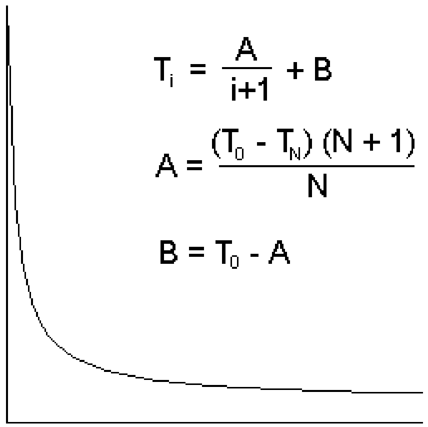
1. $P(t) = 1 - \delta / t \; ; \; \delta < t$
2. Discrete options

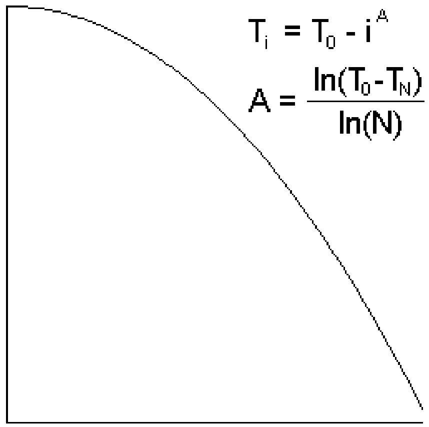
Iterations 1~100:	0.8
Iterations 101~200:	0.6
Iterations 201~300:	0.4

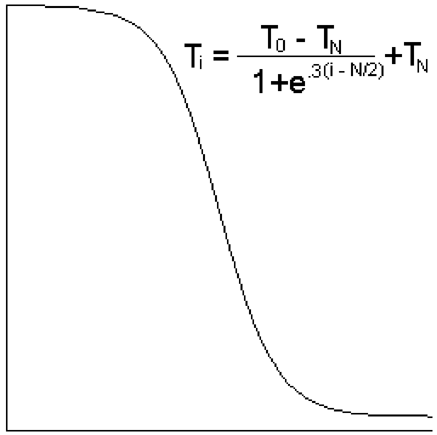
Variant: Alternative Cooling Schedules

- T_i is the temperature for step *i*, where *i* increases to N
- The initial and final temperatures, T_0 and T_N



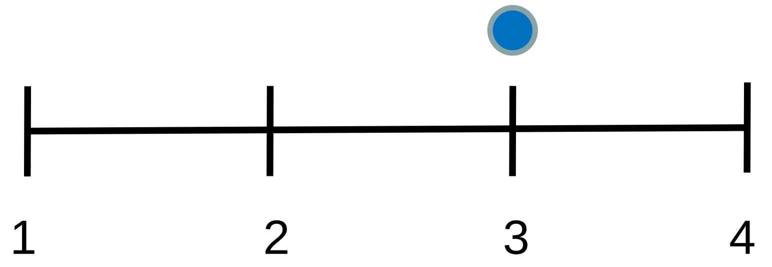






Variant: Neighborhood (1)

- Neighborhood definition is implicit, not shown in the algorithm
- Real-valued neighborhood: analogous to real-valued mutation
- Discrete neighborhood:
  - probability of choosing the next solution may be in inverse relation to the distance from the current solution



Variant: Neighborhood (2)

- Permutation neighborhood:
  - Inversion

[1 3 4 2 6 5] [1 4 3 2 6 5]

  - Transposition

[1 3 4 2 6 5] [1 6 4 2 3 5]

  - Displacement

[1 3 4 2 6 5] [1 3 6 4 2 5]

  - 2-opt (Croes, 1958)

[1 3 4 2 6 5] [1 6 2 4 3 5]

Variant: Fixed Temperature

- Simulated annealing does most of its work during cooling stage
- Annealing at a constant temperature
  - Searching for the optimum temperature, which determines the acceptance probability
  - Use the acceptance probability throughout the process

Variant: Reheating

- When a move is accepted, cool the system by $t = t / (1+\beta)$; β is a constant >0
- When a move is rejected, heat the system by $t = t / (1-\gamma)$; γ is a constant >0

following example, β=0.2 and γ =0.1, original string=[10011]

T(k)	Selected bit	String	x	f	δ	Move Prob.	Move?	String
500	1	00011	3	2287	-112	0.799315	YES	00011
416.7	3	00111	7	3803	>0		YES	00111
347.2	5	00110	6	3556	-247	0.543418	NO	00111
385.8	2	01111	15	3475	-328	0.427340	YES	01111

416.7/(1+0.2)

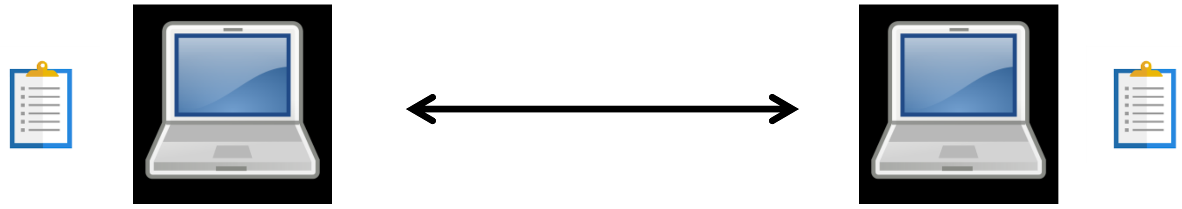
347.2/(1-0.1)

Variant: Objective Function

- The objective function is calculated at every iteration of the algorithm
- If calculation is expensive (e.g., simulation, finite element method)
  - Approximate the objective value (e.g., using fewer simulation runs); focus only on potentially good solutions
  - Evaluate just the changes and update the previous objective value
  - Keep cache to store objective values that have already been evaluated; may need a large memory space

Parallel Implementation

- Independent runs on different computing processors, with limited communication
- Split the search space into parts, each to be assigned to a particular processor
- Each processor checks the solution and test for acceptance independently. An accepted solution is conveyed to all the other processors



Hybrid with Other Algorithms

- Start with a solution that has been heuristically built by greedy algorithm or other meta-heuristics, such as GA or PSO
- Switch between different algorithms iteratively; usually use GA or PSO for global search and SA for local search
- Use SA to fine tune the control parameters of other algorithms

  GA: crossover rate, mutation rate

  PSO: w, c_1, c_2, and V_{max}

Summary of SA

- Individual-based, not population-based
- Simple in coding
- Use a nested loop structure
- No memory of past experience
- More complicated to fine tune the algorithm; particularly decreasing cooling schedule and increasing function of iterations