Trajectory-based Optimization:

Simulated Annealing - I

Lecture 7 – Thursday May 29, 2014

Outline

Search Methods

Deterministic

Numerical Methods

Graph Search Methods

Blind Search
- ◇ Depth-first
- ◇ Breadth-first
- ◇ Dijkstra

Informed Search
- ◇ Hill climbing
- ◇ Beam search
- ◇ Best-first, A*

Game Playing as search

Stochastic

Trajectory Methods
- ◇ TS
- ◇ **SA**

Population-based Methods

Evolutionary Computing
- ◇ **GA**

Swarm Intelligence
- ◇ **PSO**
- ◇ **ACO**

Outline

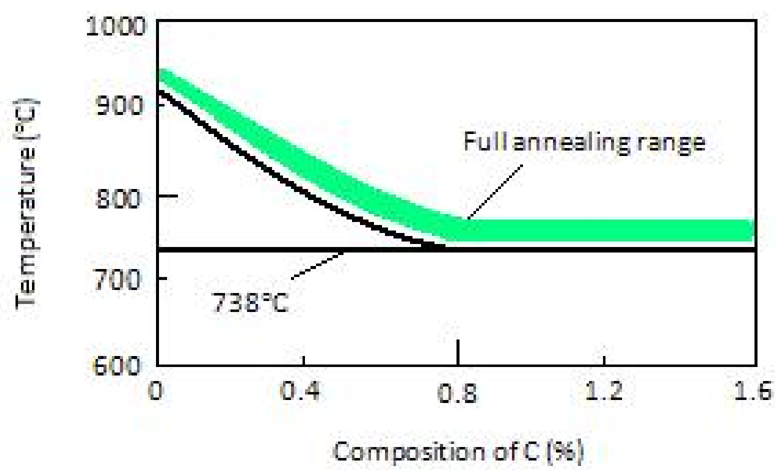
- Physical Annealing
- Simulated Annealing
- SA Cooling Schedule
- SA for TSP
- SA for PLP
- SA for Scheduling problems
- SA for Function Optimization
- Adaptive SA
- Cooperative SA
- Summary

Outline

- **Physical Annealing**
- Simulated Annealing
- SA Cooling Schedule
- SA for TSP
- SA for PLP
- SA for Scheduling problems
- SA for Function Optimization
- Adaptive SA
- Cooperative SA
- Summary

Physical Annealing

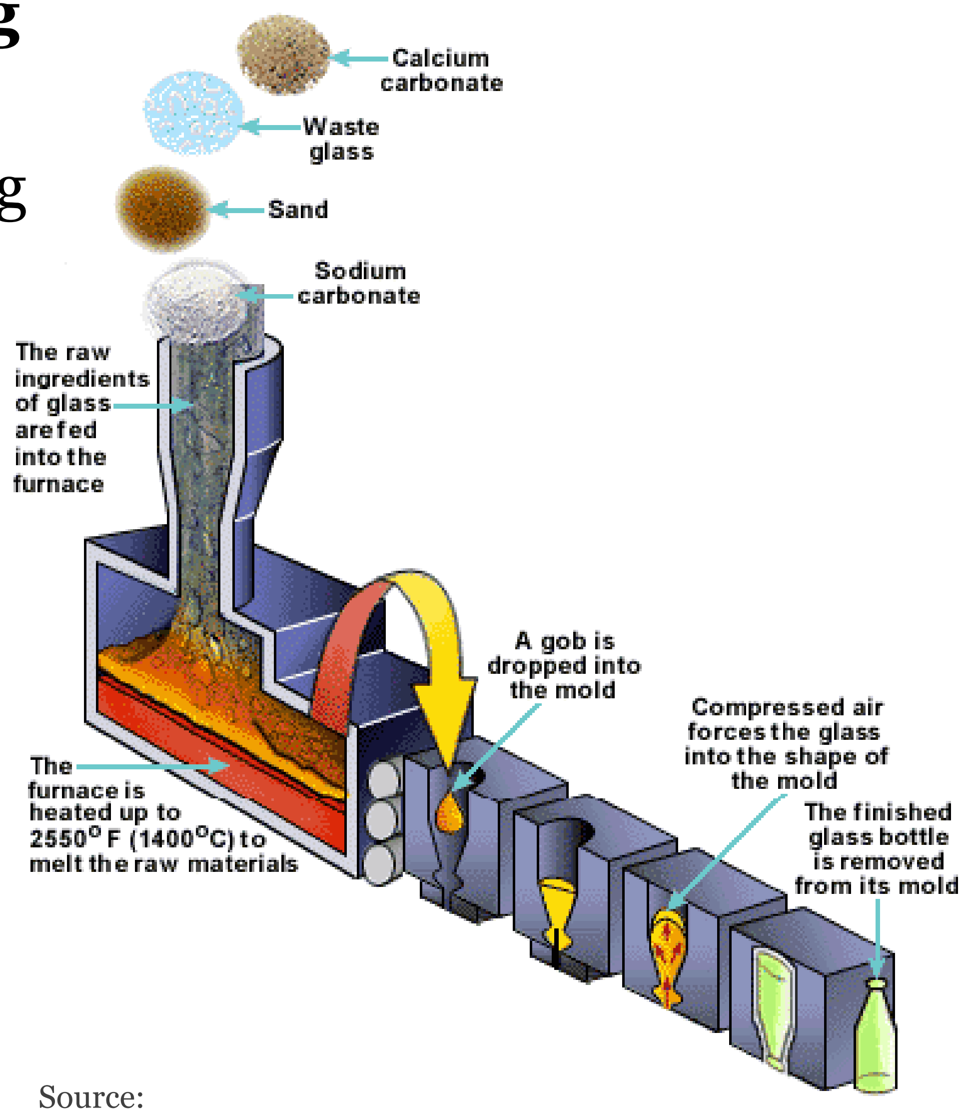
- **Annealing** process has been used since **5000 B.C.** and mainly used with glass and crystals.
- **Annealing** is a **heat treatment** wherein a material is altered, causing **changes in its properties** such as **strength** and **hardness**.
- It is a process that produces conditions by heating to above the **recrystallization temperature** and maintaining a suitable temperature, and then cooling



Physical Annealing

- **Annealing in Bottle Making**

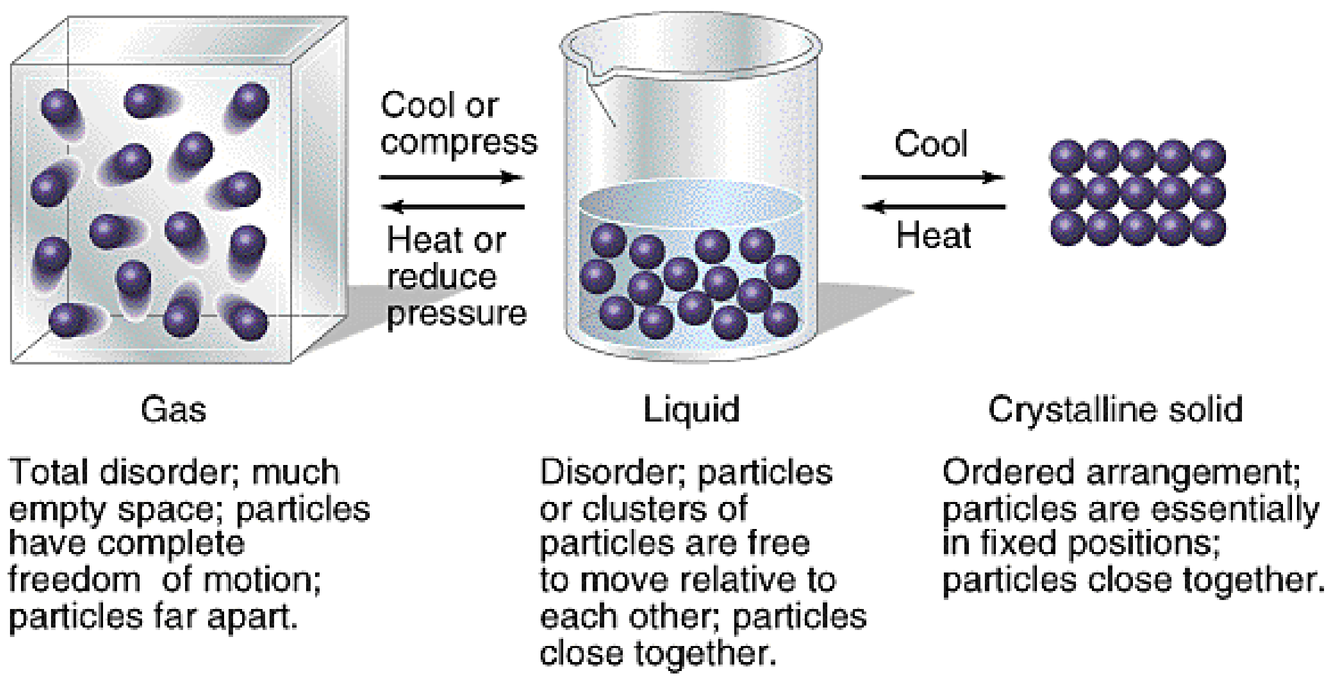
  Annealing is done by reheating the glass and **gradually** cooling it. Such a process removes the stresses and strains in the glass after shaping. This is an important step and if not done may cause the glass to shatter as a result of the build up of tension caused by uneven cooling. After the bottles have cooled to room temperature, they are inspected and finally packaged.



Source:
http://www.wisedude.com/science_engineering/bottles.htm

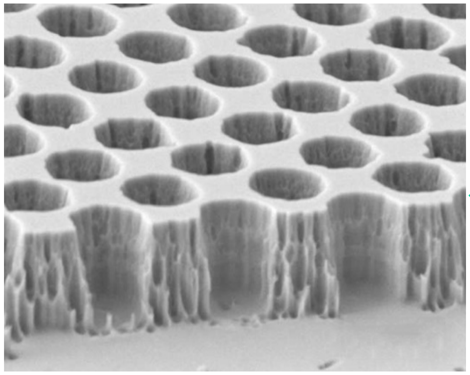
Physical Annealing

- As **temperature reduces**, the **mobility of molecules reduces**, with the tendency that molecules may **align** themselves in a **crystalline structure**.
- The aligned structure is the **minimum energy state** for the system.

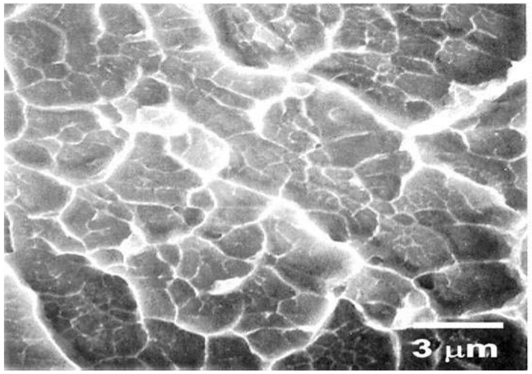


Physical Annealing

- To ensure that this **alignment** is obtained, cooling must occur at a **sufficiently slow rate**.
- If the substance is cooled at a **too rapid rate**, an **amorphous state** may be reached.



A metal with crystalline structure



Amorphous metal with a disordered atomic scale structure

Physical Annealing

- The annealing process involves the **careful control** of temperature and cooling rate, often called **annealing or cooling schedule**.
- The annealing schedule is very critical:
  - ◊ **Annealing times** should be **long enough** for the material to undergo the required transformation,
  - ◊ If the difference in the **temperatures rate of change** between the outside and inside of a material is **too big**, this may cause **defects and cracks**.

Outline

- Physical Annealing
- **Simulated Annealing**
- SA Cooling Schedule
- SA for TSP
- SA for PLP
- SA for Scheduling problems
- SA for Function Optimization
- Adaptive SA
- Cooperative SA
- Summary

Simulated Annealing

- Simulated annealing is an optimization process based on the physical annealing process.
- In the context of mathematical optimization, the **minimum of an objective function** represents the **minimum energy** of the system.
- Simulated annealing is an algorithmic implementation of the cooling process to find the **optimum of an objective function**.

Simulated Annealing

Physical Annealing	Simulated Annealing
State of a system	Solution of a problem
Energy of a state	Cost of a solution
Temperature	Control parameter (temperature)
Frozen state	Final solution
Molecules move freely	Explore parameter space
Molecules are stuck	Restrict exploration

Simulated Annealing

- The first SA algorithm was developed in 1953 (Metropolis) [1].
- Kirkpatrick (1982) applied SA to optimization problems [2].
- It was used for locating a good approximation of the global optimum in large search spaces.
- **Applications:**
  - ◇ Non-linear function optimization.
  - ◇ Travelling Salesman Problem (TSP).
  - ◇ Academic course scheduling.
  - ◇ Network design.
  - ◇ Task allocation.
  - ◇ Circuit partitioning and placement.
  - ◇ Strategy scheduling for capital products with complex product structure.
  - ◇ Umpire scheduling in US Open Tennis tournament! and more…

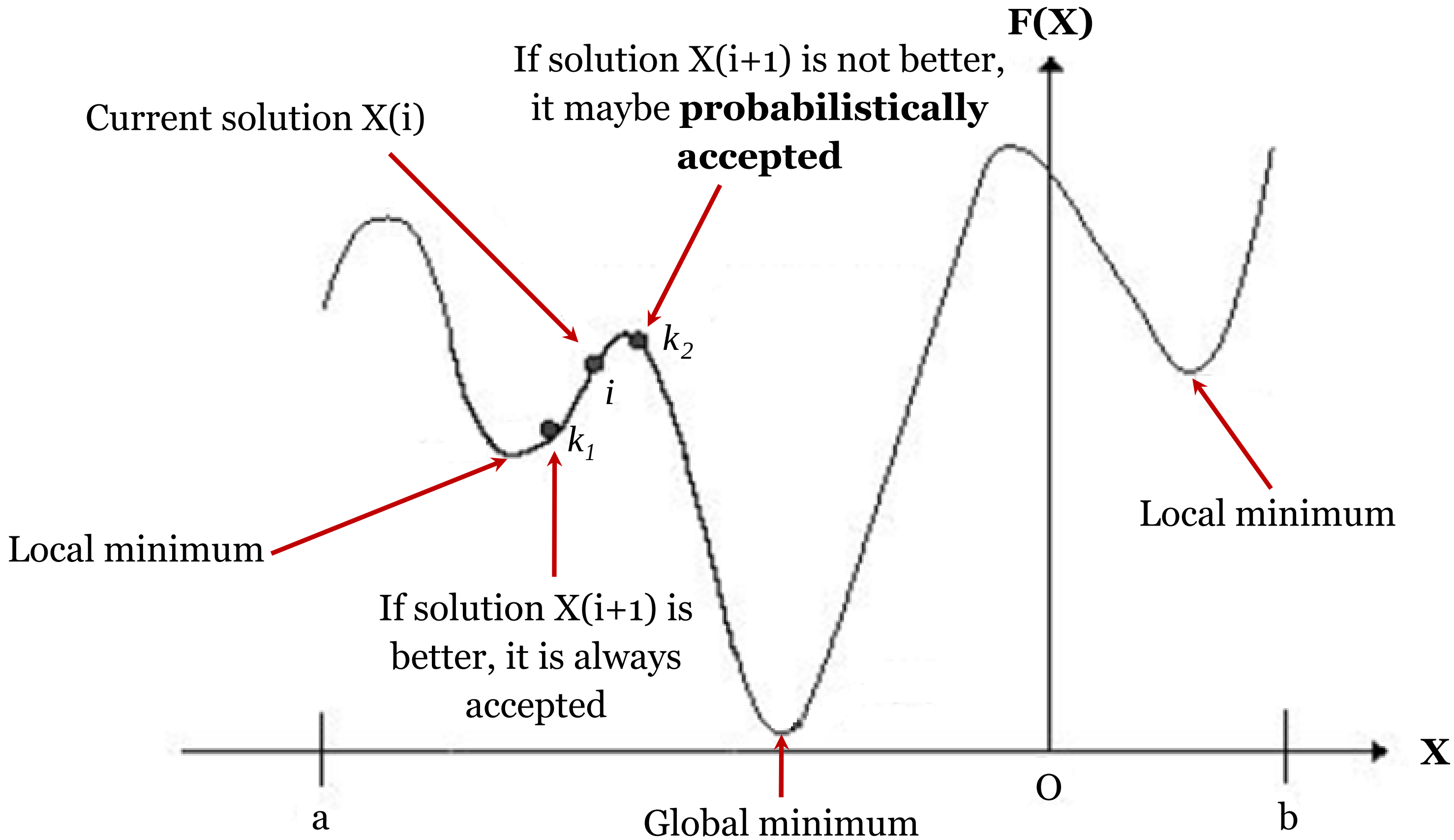
Simulated Annealing

Advantages	Disadvantages
Ease of use	A lot of computer time for many runs.
Providing good solutions for a wide range of problems.	A lot of tuneable parameters.

[3]

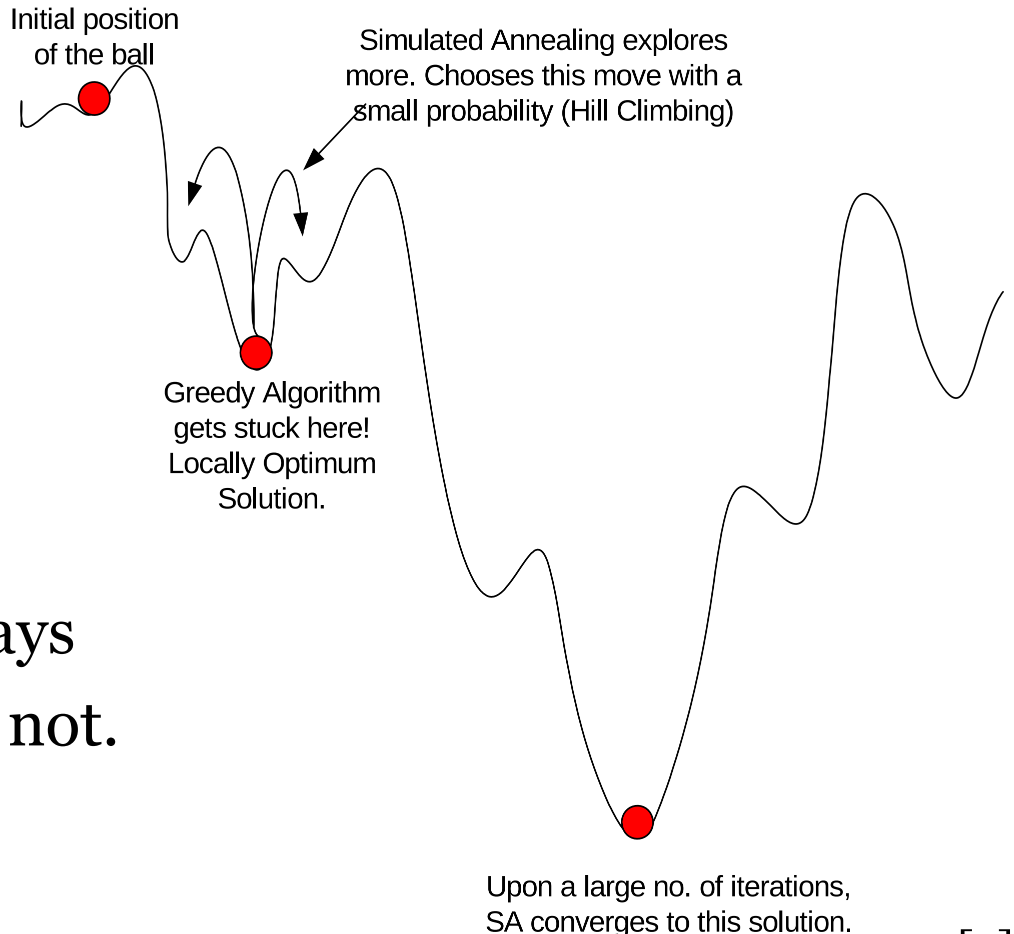
Simulated Annealing

SA uses a **random search strategy**, which not only accepts new positions that decrease the objective function (assuming a minimization problem), but **also accepts positions that increase objective function values**.



Simulated Annealing

- **SA vs. Hill Climbing:**
  - ◊ Compared to hill climbing the main difference is that **SA probabilistically allows downwards steps** controlled by current temperature and how bad move is.
  - ◊ In SA better moves are always accepted. Worse moves are not.

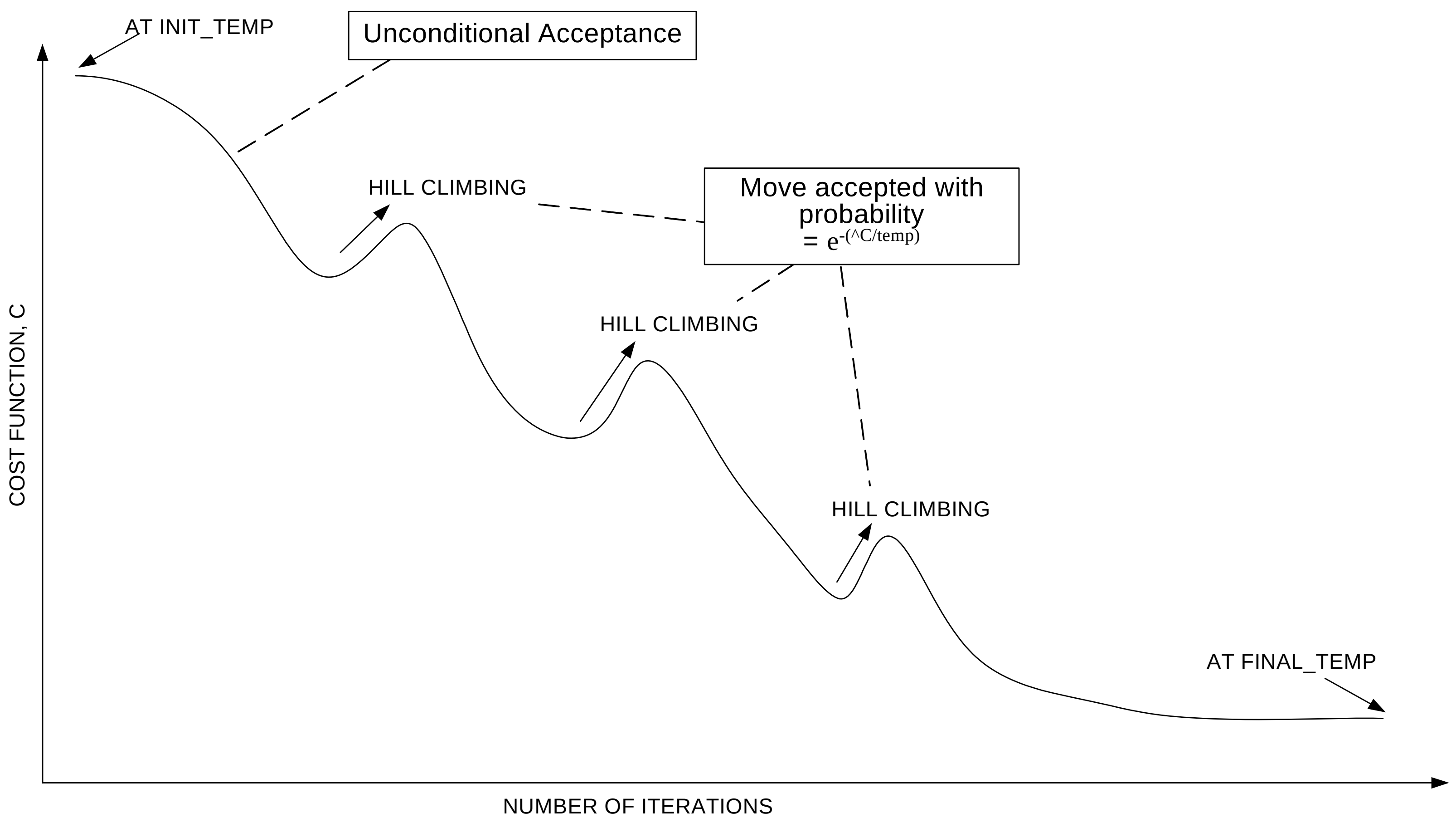


[4]

Simulated Annealing

- **To accept or not to accept?**

The move is accepted **probabilistically** based on the **Boltzmann–Gibbs** distribution.



[4]

Simulated Annealing

- **To accept or not to accept?**

In thermodynamics, a state at a temperature **t** has a probability of an increase in the energy magnitude **ΔE** given by **Boltzmann–Gibbs** distribution as follows:

$$p(\Delta E) = e^{-\Delta E/(k \times t)}$$

where k is the Boltzman constant.

Simulated Annealing

- **To accept or not to accept?**

The simplest way to link **ΔE** with the change of the objective function ***Δf*** is to use:

$$\Delta E = \gamma \Delta f$$

where γ is a real constant.

For simplicity without losing generality, we can use k=1 and γ=1.

Thus, the probability ***p*** simply becomes:

$$p(\Delta f, T) = e^{-\Delta f / T}$$

Whether or not we accept a change, we usually use a random number *r* in the interval [0,1] as a threshold.

Thus, if $p > r$ *or* $p = e^{-\Delta f / T} > r$ the move is accepted.

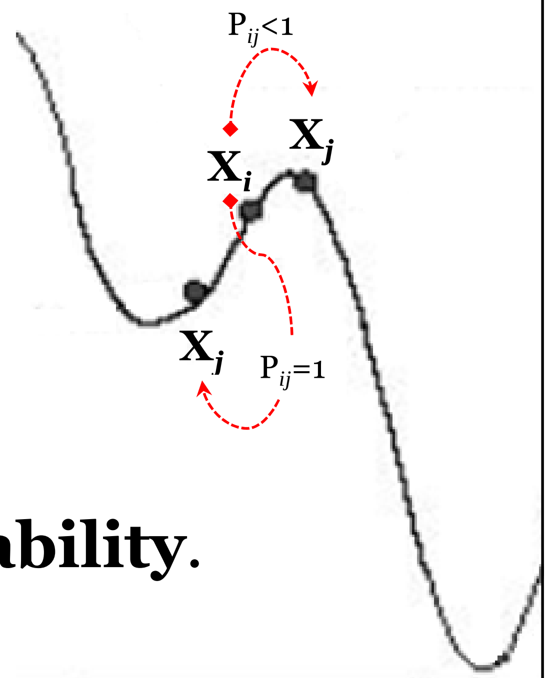
Simulated Annealing

- **To accept or not to accept?**

If $\boldsymbol{P_{ij}}$ is the *probability* of moving from point $\mathbf{x}_i$ to $\boldsymbol{x}_j$,

then P_{ij} is calculated using:

$$P_{ij} = \begin{cases} 1 & \text{if } f(\mathbf{x}_j) < f(\mathbf{x}_i) \\ e^{-\frac{f(\mathbf{x}_j) - f(\mathbf{x}_i)}{T}} & \text{otherwise} \end{cases}$$



where T is the temperature of the system.

The probability $\boldsymbol{P_{ij}}$ is called **transition probability**.

Simulated annealing = stochastic hill climbing with dynamic parameter T

Simulated Annealing

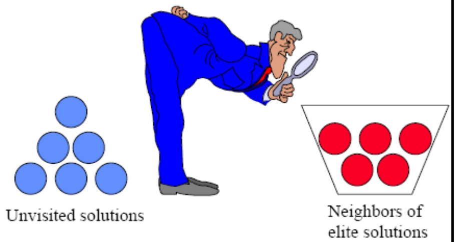
- **To accept or not to accept?**

$$P_{ij} = \begin{cases} 1 & \text{if } f(\mathbf{x}_j) < f(\mathbf{x}_i) \\ e^{-\frac{f(\mathbf{x}_j)-f(\mathbf{x}_i)}{T}} & \text{otherwise} \end{cases}$$

Change	Temperature	Acceptance/Transition Probability
0.2	0.95	0.810157735
0.4	0.95	0.656355555
0.6	0.95	0.53175153
0.8	0.95	0.430802615
0.2	0.1	0.135335283
0.4	0.1	0.018315639
0.6	0.1	0.002478752
0.8	0.1	0.000335463

◇ At high temperatures, explore parameter space.

◇ At lower temperatures, restrict exploration.



Simulated Annealing

- **To accept or not to accept?**

$$P_{ij} = \begin{cases} 1 & \text{if } f(\mathbf{x}_j) < f(\mathbf{x}_i) \\ e^{-\frac{f(\mathbf{x}_j) - f(\mathbf{x}_i)}{T}} & \text{otherwise} \end{cases}$$

◇ The **probability of accepting** a worse state is a function of both the **temperature** of the system and the **change in the cost function**.

◇ As the **temperature decreases**, the **probability** of accepting worse moves **decreases**.

◇ Q: When is SA converted into hill climbing?

A: If T=0, **no worse moves are accepted**.

[5]

Simulated Annealing

- **SA Algorithm**

Objective function $f(x)$, $x=(x_1,...,x_p)^T$

Initialize initial temperature T_o and initial guess $x^{(o)}$

Set final temperature T_f nd max number of iterations N

Define cooling schedule

While ($T> T_f$ and $n<N$)

    Move randomly to new locations: $x_{n+1}=x_n$+randn

    Calculate $\Delta f=f_{n+1}(x_{n+1}) - f_n(x_n)$

    Accept the new solution if better

    if no improved

        Generate a random number r

        Accept if $p=\exp(-\Delta f/T) > r$

    end if

    Update the best x^* and f^*

    $n=n+1$

End while

Outline

- Physical Annealing
- Simulated Annealing
- **SA Cooling Schedule**
- SA for TSP
- SA for PLP
- SA for Scheduling problems
- SA for Function Optimization
- Adaptive SA
- Cooperative SA
- Summary

SA Cooling Schedule

Cooling Schedule

Starting Temperature + Final Temperature + Decrement Temperature + Iterations at each temperature

SA Cooling Schedule

- **Starting Temperature**
  - ◇ The choice of the right initial temperature is crucially important.

$$P_{ij} = \begin{cases} 1 & \text{if } f(\mathbf{x}_j) < f(\mathbf{x}_i) \\ e^{-\frac{f(\mathbf{x}_j) - f(\mathbf{x}_i)}{T}} & \text{otherwise} \end{cases}$$

  - ◇ For a given change Δf, if **T is too high (T → ∞)**, then ***p*** **→ 1**, which means almost **all the changes** will be **accepted**.
  - ◇ If T **is too low (T → 0)**, then any $\Delta f > 0$ (**worse solution**) will rarely be accepted as **p → 0** and thus the **diversity** of the solution is **limited**, but any improvement Δf will almost always be accepted (may be **trapped in local minima**).

SA Cooling Schedule

- **Starting Temperature**
  - ◊ In order to find a **suitable starting temperature** T_o, we can use any available information about the objective function.
  - ◊ If we know the **maximum change max(Δ*f*)** of the objective function, we can use this to estimate an initial temperature T_o for **a given probability *p_o***.

$$T_o \approx -\frac{\max(\Delta f)}{\ln p_o}$$

SA Cooling Schedule

- **Starting Temperature**
  - ◊ If we do not know the possible maximum change of the objective function, we can use a **heuristic approach**.
  - ◊ We can:
    - – **start evaluations from a very high temperature** (so that almost all changes are accepted) and
    - – reduce the temperature quickly until about **50% or 60% of the worse moves are accepted**, and
    - – then **use this temperature** as the new initial temperature T_o for proper and relatively slow cooling processing.

SA Cooling Schedule

- **Final Temperature**
  - ◊ It is usual to let the temperature decrease until it reaches **zero** However, this can make the algorithm run for **a lot longer**, especially when a geometric cooling schedule is being used.
  - ◊ In practise, it is **not necessary** to let the temperature reach zero because the chances of accepting a worse move are almost the same as the temperature being equal to zero.
  - ◊ Therefore, the stopping criteria can either:
    - be a suitably low temperature ($\mathbf{T_f = 10^{-10} \sim 10^{-5}}$) or
    - when the system is **"frozen"** at the current temperature (i.e. no better or worse moves are being accepted).

SA Cooling Schedule

- **Temperature Decrement or Annealing Schedule**

  Two commonly used annealing schedules (or cooling schedules) are:

  ◊ **Linear cooling schedule:** $T = T_o - \beta i$

  where T_o is the initial temperature, and

  i is the pseudo time for iterations,

  β is the cooling rate, and it should be chosen in such a way that

  $T \rightarrow 0$ when $i \rightarrow i_f$ (or the maximum number N of iterations), this usually gives:

$$\beta = \frac{(T_o - T_f)}{i_f}$$

SA Cooling Schedule

- **Temperature Decrement or Annealing Schedule**

  ◇ **Geometric cooling schedule:**

  A geometric cooling schedule essentially decreases the temperature by a cooling factor $0 < \alpha < 1$ so that T is replaced by αT or

$$T(t) = T_o \alpha^i, \qquad i = 1, 2, \ldots, i_f$$

  The cooling process should be slow enough to allow the system to stabilize easily.

  In practice, α = **0.7 ~ 0.95** is commonly used.

  The **higher** the value of α, the **longer** it will take to reach the final (low) temperature.

SA Cooling Schedule

- **Temperature Decrement or Annealing Schedule**

  ◊ **Linear vs. Geometric cooling**

  Linear Cooling: $T = T_o - \beta i,$ $\beta = \frac{(T_o - T_f)}{i_f}$

  Geometric Cooling: $T(t) = T_o \alpha^i,$ $i = 1,2,\ldots,i_f$

  The advantage of the geometric method is that:

  $T \rightarrow 0$ when $i \rightarrow \infty$, and thus there is **no need to specify the maximum number of iterations**.

SA Cooling Schedule

- **Iterations at each temperature**
  - ◇ Enough iterations should be allowed at every temperature for the system to be **stable** at that temperature.
  - ◇ The required **number of iterations** might be **exponential to the problem size**.
  - ◇ Usually, a **constant value** is used.
  - ◇ We could dynamically change this value:
    - – Allowing a **small number** of iterations at **high temperatures**
    - – Allowing a **large number of iterations** at **low temperature** to fully explore the local optimum.

Outline

- Physical Annealing
- Simulated Annealing
- SA Cooling Schedule
- **SA for TSP**
- SA for PLP
- SA for Scheduling problems
- SA for Function Optimization
- Adaptive SA
- Cooperative SA
- Summary

SA for TSP

- **Problem**

  ◇ Given **n** cities,

  ◇ A travelling salesman must visit the n cities and return home, making a loop (roundtrip).

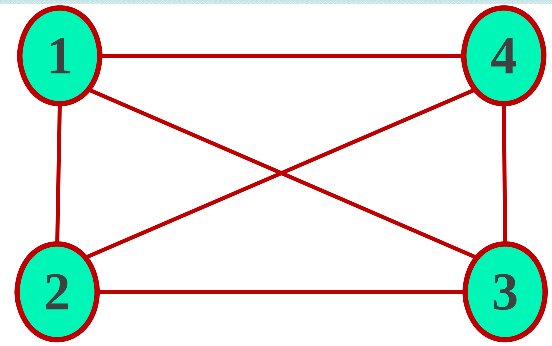
  ◇ He would like to travel in the most efficient way (cheapest way or shortest distance or some other criterion).

- **Search Space**

  Search space is BIG:

  for 21 cities there are 21! ≈ 51,090,942,171,709,440,000 **possible** tours. This is NP-Hard problem.

SA for TSP



TSP

Symmetric

In the symmetric TSP, the **distance between two cities is the same in each opposite direction**, forming an **undirected graph**. This symmetry halves the number of possible solutions.

#routes=4!/2=12

Asymmetric

In the asymmetric TSP, **paths may not exist in both directions** or the **distances might be different**, forming a **directed graph**. Traffic collisions, one-way streets, and airfares for cities with different departure and arrival fees are examples of how this symmetry could break down.

#routes=4!=24

SA for TSP

TSP Applications

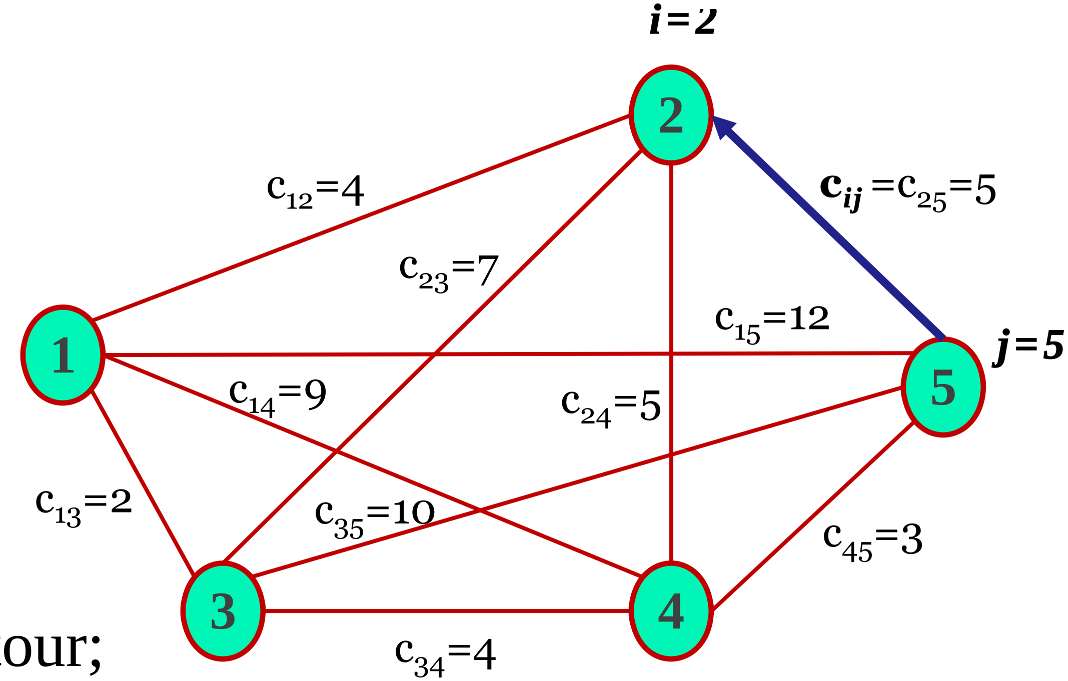
TSP is used as a **platform** for the study of general methods that can be applied to a wide range of **discrete optimization** problems

- **Microchips manufacturing:** manufacture of a circuit board, it is important to determine the best order in which a laser will drill thousands of holes.
- **Assignment of routes for planes of a specified fleet**
- **Permutation flow shop scheduling problem (PFSP):** consists of finding a sequence for the jobs that minimizes the makespan; that is, the overall time to complete the schedule.
- **Transportation and Logistics**: problem of arranging school bus routes to pick up the children in a school district, transportation of farming equipment from one location to another to test soil or scheduling of service calls at cable firms, the delivery of meals to homebound persons, the scheduling of stacker cranes in warehouses, the routing of trucks for parcel post pickup, and a host of others.
- **Overhauling of gas turbine engines**
- **For more:** http://iris.gmu.edu/~khoffman/papers/trav_salesman.html

SA for TSP

- **TSP Formulation**
  - ◇ **Miller–Tucker–Zemlin (MTZ) Problem Formulation**

In the complete directed graph $D=(N,A)$, with arc-costs c_{ij} , we seek the tour (a directed cycle that contains all n cities) of minimal length.



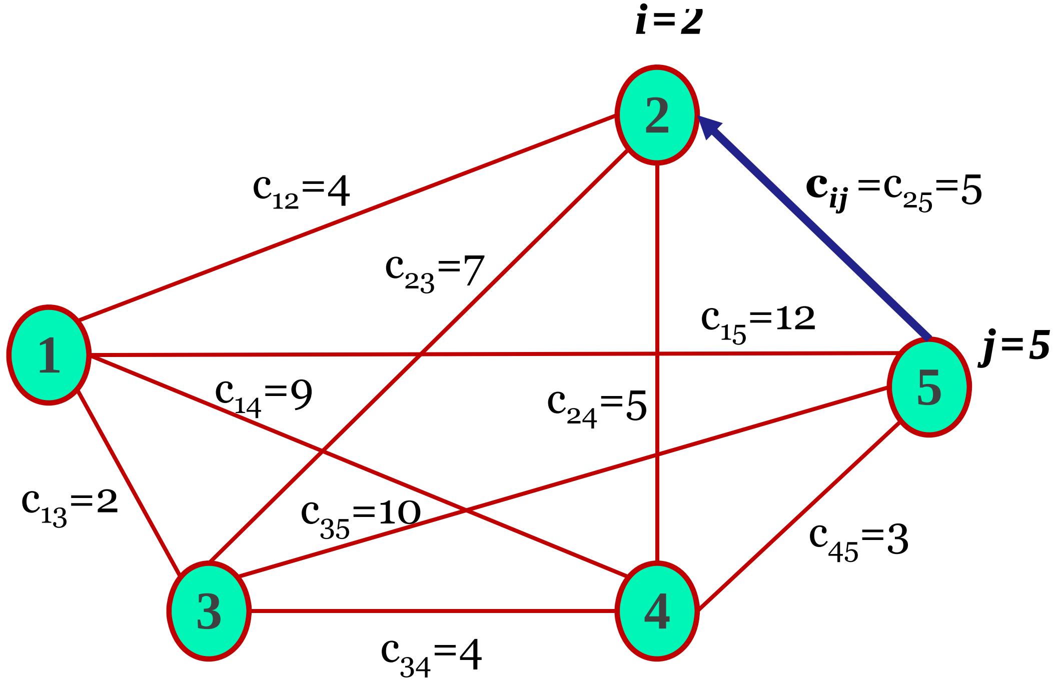
$$x_{ij} = \begin{cases} 1 & \text{if arc(i, j) is in the tour;} \\ 0 & \text{otherwise} \end{cases}$$

[6]

SA for TSP

- **TSP Formulation**
  - ◇ **Miller–Tucker–Zemlin (MTZ) Problem Formulation**

The TSP can be defined as a minimization problem as follows:



$$\min \quad \sum_{i,j} c_{ij} x_{ij}$$

Subject to two types of constraints:

1. Assignment Constraints

$$\sum_i x_{ij} = 1 \quad \forall i, \qquad \sum_j x_{ij} = 1 \quad \forall j,$$

$$0 \le x_{ij} \le 1 \qquad x_{ij} \quad \text{integer},$$

[6]

SA for TSP

- **TSP Formulation**
  - **Miller–Tucker–Zemlin (MTZ) Problem Formulation**

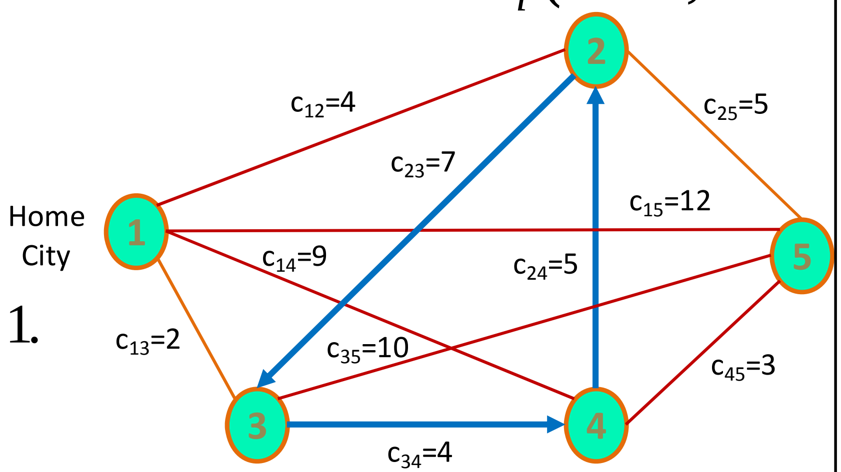
2. Subtour (or subtour elimination) Constraints

To exclude subtours, one can use extra variables u_i $(i = 1, \ldots, n)$, and the constraints

$$u_1 = 1,$$

$$2 \leq u_i \leq n \qquad \forall i \neq 1,$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}) \qquad \forall i \neq 1, \forall j \neq 1.$$



Arc	Constraint	Validity
2-3	2-3+1≤(4-1)(1-1)	√
3-4	3-4+1≤(4-1)(1-1)	√
4-2	4-2+1≤(4-1)(1-1)	X

SA for TSP

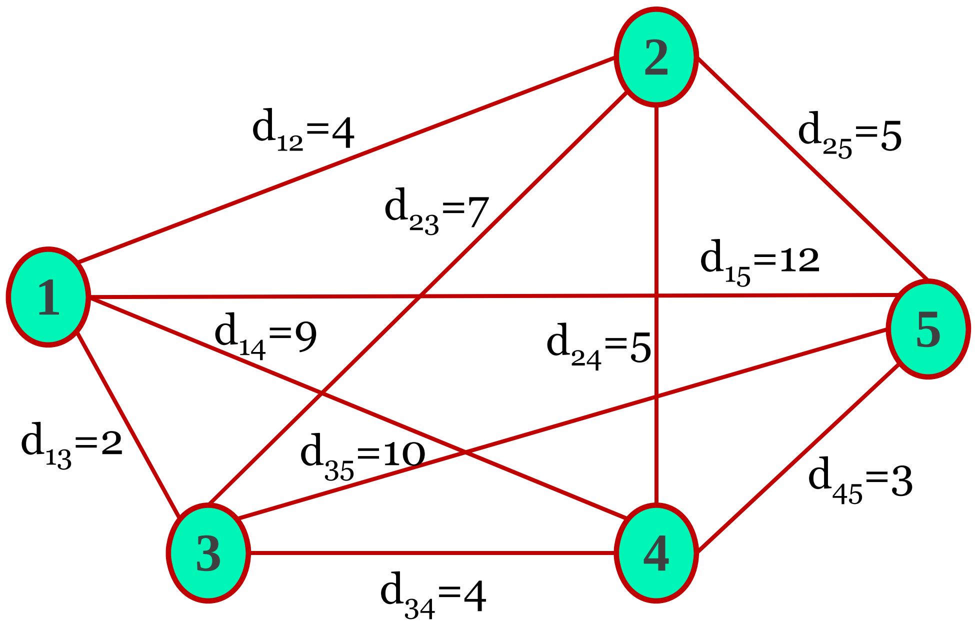
- The solution would be a **permutation** of n cities:

$$Sol = [\ 1, 2, 3, .., n\ ]$$

- We should define a **neighborhood** for the solution in order to generate the next one.
- The new solution could be generated by performing a predetermined **number of swaps** on the current solution.
- Some **adaptive implementations** decrease the number of cities to swap during the run. Thus **reducing the neighborhood size** as the **search progresses**.

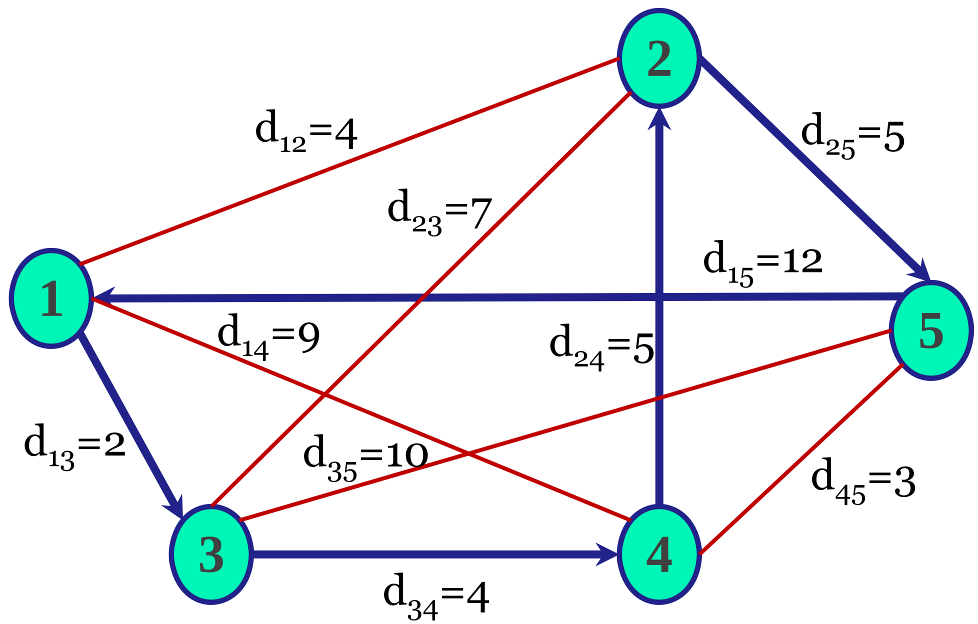
SA for TSP

- ***Example:*** 5 cities – 120/2=60 possible tours (symmetric TSP)



SA for TSP

- Start with a **random** solution: Sol = [1, 3, 4, 2, 5]



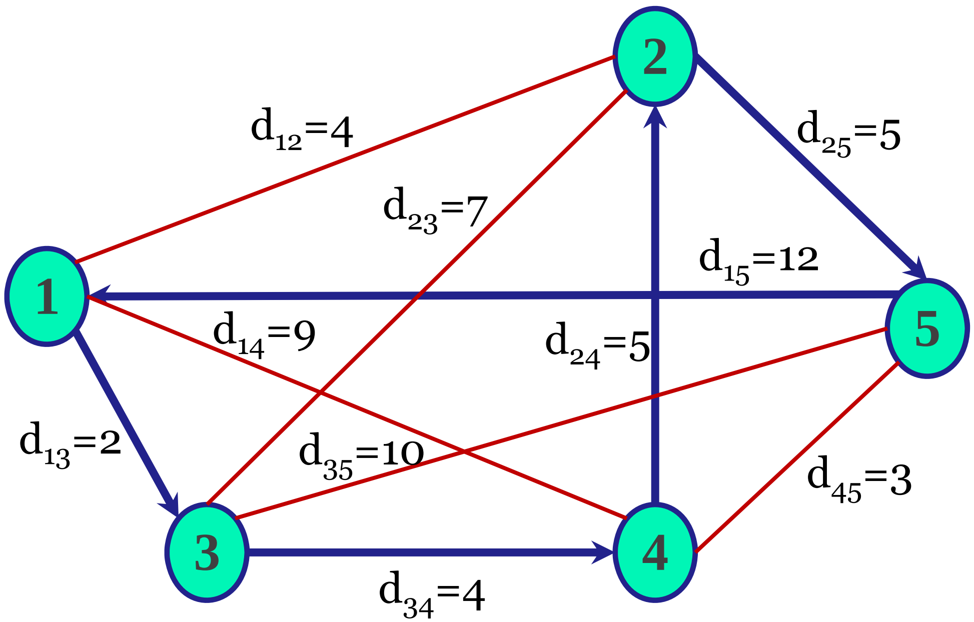
- The total distance would be:

For a closed tour

$$total\_dist = \sum_{i=1}^{n-1} d_{sol_i} d_{sol_{i+1}} + d_{sol_n} d_{sol_1}$$

SA for TSP

- Start with a **random** solution: Sol = [1, 3, 4, 2, 5]



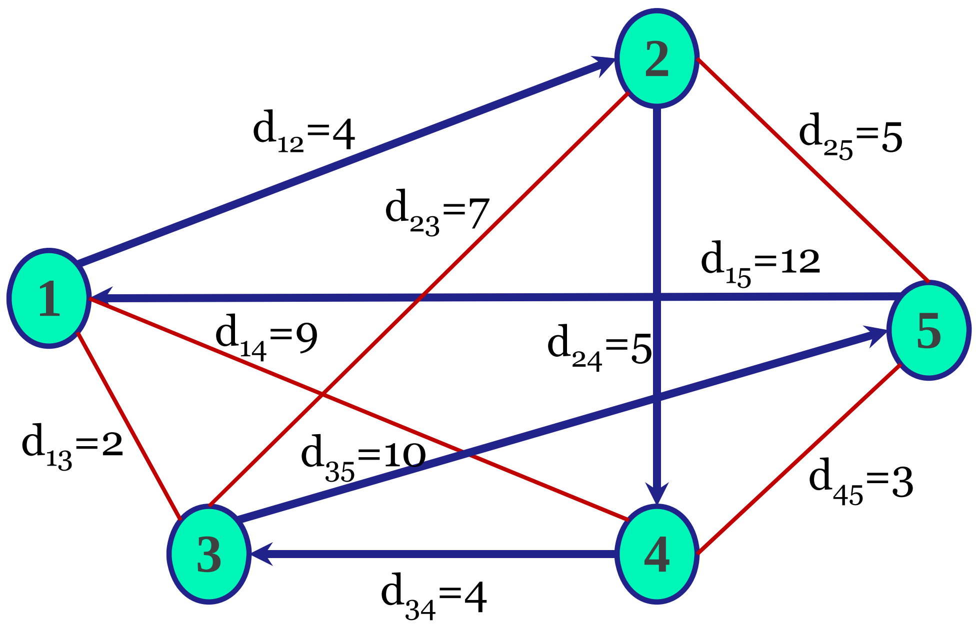
- The tour length of the initial solution is:

$$total\_dist = 2 + 4 + 5 + 5 + 12 = 28$$

SA for TSP

- To generate a candidate solution, select two random cities and swap them:

Sol = [1, 2, 4, 3, 5]



- The tour length of the candidate solution is:

$$total\_dist = 4 + 5 + 4 + 10 + 12 = 35$$

SA for TSP

- Initial solution: Sol = [1, 3, 4, 2, 5] Cost=28
- New solution: Sol = [1, 2, 4, 3, 5] Cost=35
- Since the new solution has a **longer tour length**, it will be **conditionally accepted** according to a **probability** of (at higher temperatures, there's a higher probability of acceptance):

$$p = e^{-\Delta f/T} = e^{-(35-28)/T} = e^{-7/T}$$

◇ Pick a random value ***r*** within 0 and 1:

◇ If ***P>r***, accept this solution

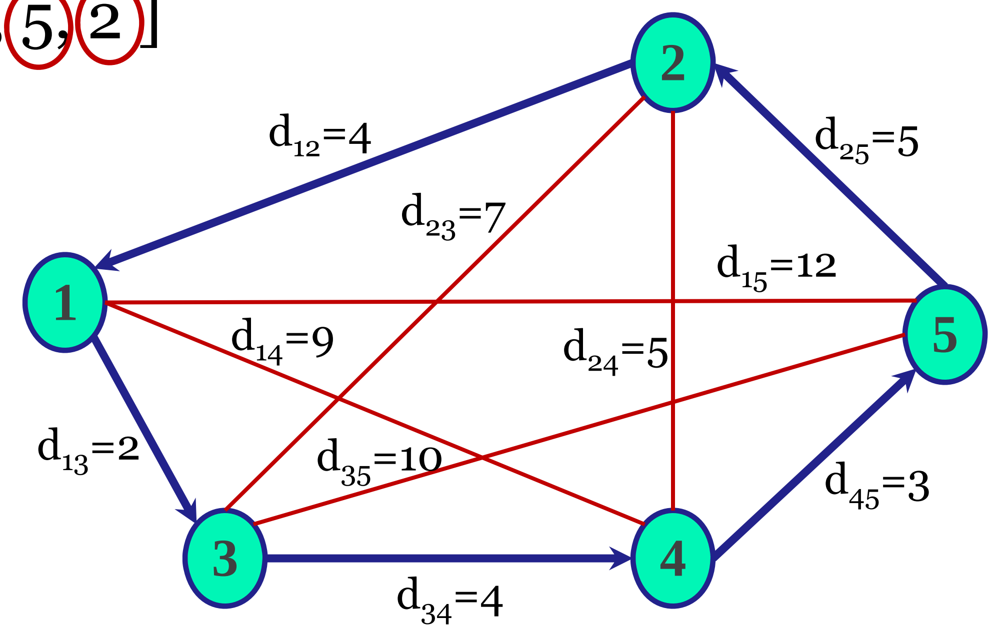
◇ **Otherwise** reject this solution

SA for TSP

- Assuming the new solution **was not accepted**,

  we generate a different one starting from the initial solution:

Sol = [1, 3, 4, 5, 2]



- The tour length of the candidate solution is:

$$total\_dist = 2+4+3+5+4=18$$

SA for TSP

- Since this solution has a shorter tour length, it will get accepted.
- The search continues …

SA for TSP

- The SA was applied for the **berlin52 TSP** * instance by having:
  - ◇ T_initial = 10000,
  - ◇ T_final = 0.1,
  - ◇ α = 0.85,
  - ◇ Using 1000 iteration per for each T
  - ◇ Using different number of swaps for obtaining the neighboring solution (1, 5,10 and 15).

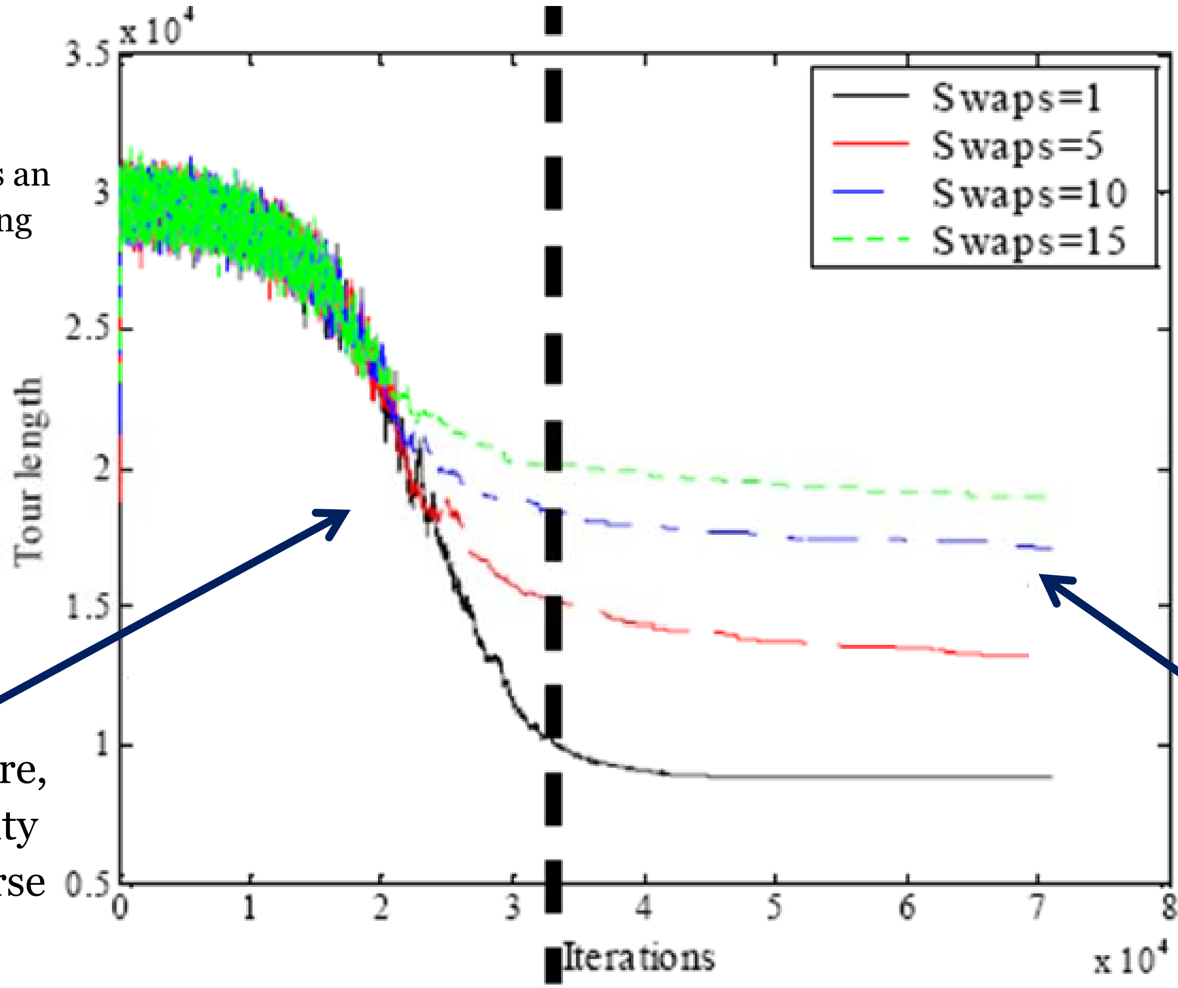
*TSPLib: http://www2.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[3]

SA for TSP

Number of swaps	1	5	10	15
Tour length	**8861**	13180	17124	18900

Green blotch here is an artifact of the plotting



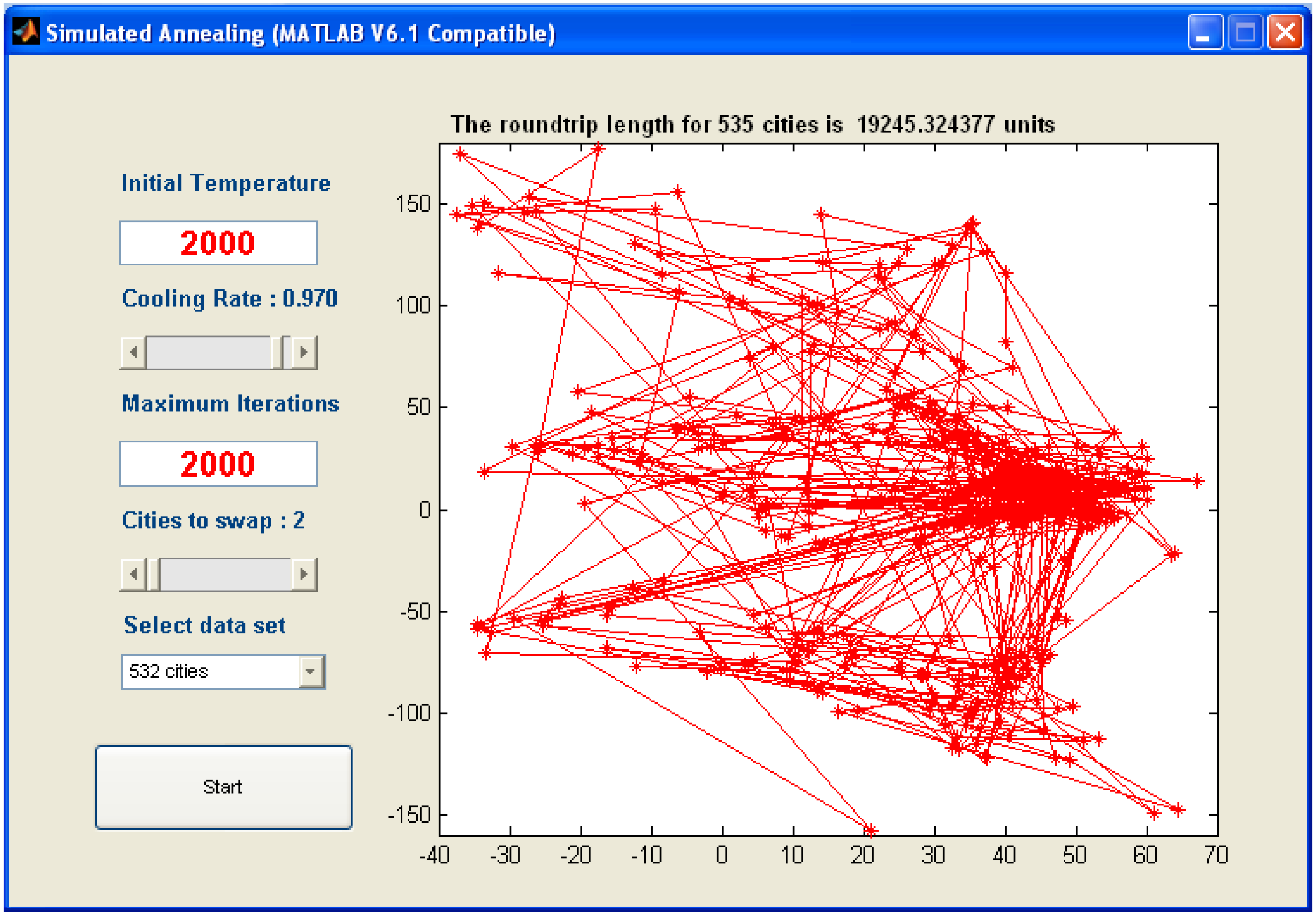
High temperature, higher probability of accepting worse solutions

Low temperature, lower probability of accepting worse solutions

[3]

SA for TSP

- **Tutorial-1: TSP using Simulated Annealing**



Resources page of the Course website

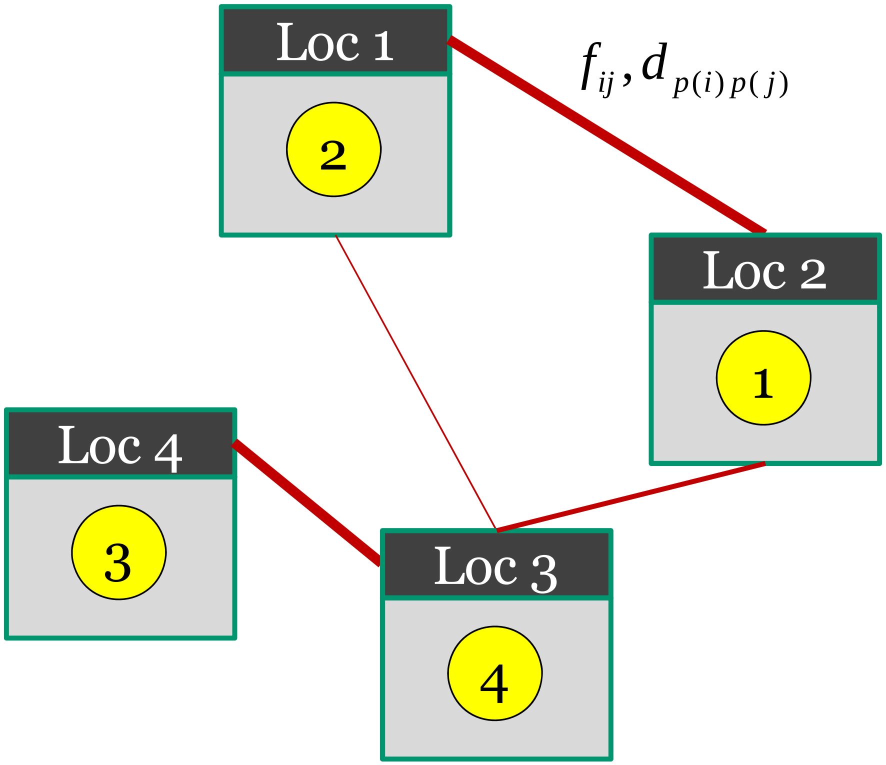
Outline

- Physical Annealing
- Simulated Annealing
- SA Cooling Schedule
- SA for TSP
- **SA for PLP**
- SA for Scheduling problems
- SA for Function Optimization
- Adaptive SA
- Cooperative SA
- Summary

SA for PLP

- **Plant Layout Problem (PLP)**

The Plant Layout Problem (PLP) aims at **assigning different facilities (departments)** to **different locations** in order to **minimize the total material handling cost**.



SA for PLP

- **Plant Layout Problem (PLP): Flow Matrix**

Assume that the following flow matrix indicates the flow of products between the different facilities.

Flow Matrix

	A	B	C	D	E	F
A	0	50	100	0	0	0
B	0	0	35	25	100	90
C	0	75	0	60	75	25
D	0	50	50	0	100	25
E	0	75	50	140	0	10
F	0	0	0	0	0	0

SA for PLP

- **Plant Layout Problem (PLP): Distance Matrix**

The **distance matrix** indicates the distances between the different locations.

Assuming that we're moving **only vertically or horizontally** (**No diagonal** move). Assume also that if we have 2 paths, we take the **shortest**. The distance matrix in this case is as follows:

A 1	B 2	C 3
D 4	E 5	F 6

	A	B	C	D	E	F
A	0	1	2	1	2	3
B	1	0	1	2	1	2
C	2	1	0	3	2	1
D	1	2	3	0	1	2
E	2	1	2	1	0	1
F	3	2	1	2	1	0

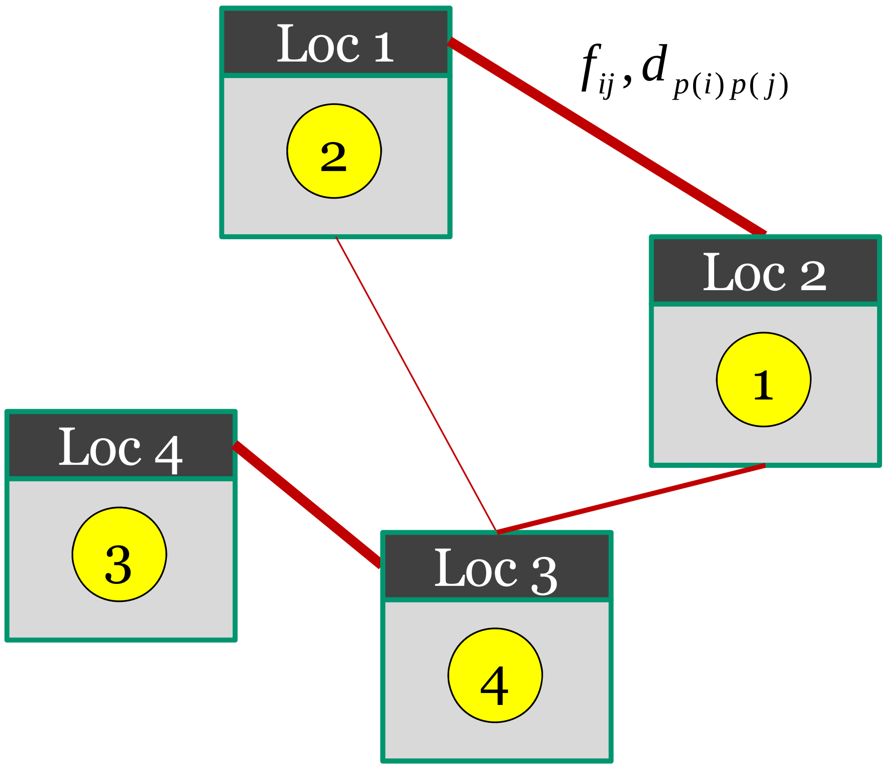
SA for PLP

- **Plant Layout Problem (PLP): Total Cost**
  - ◇ **Material Handling Cost=distance*flow**

$$MHC_{ij} = d_{ij} \times f_{ij}$$

  - ◇ TMHC is the **summation** of all the material handling costs inside the matrix.

$$TMHC = \sum MHC_{ij}$$



SA for PLP

- **Plant Layout Problem (PLP): Example**

  It is required to **allocate** the six departments to the six locations to minimize the **Total Material Handling Cost (TMHC)**.

  1. Generate an initial solution.
  2. Define a suitable neighborhood operator.
  3. Define a suitable annealing schedule and perform 2 iterations of SA.

SA for PLP

- **Initial Solution**

A 1	B 2	C 3
D 4	E 5	F 6

1	2	3	4	5	6
A	B	C	D	E	F

- **Distance Matrix**

	A	B	C	D	E	F
A	0	1	2	1	2	3
B	1	0	1	2	1	2
C	2	1	0	3	2	1
D	1	2	3	0	1	2
E	2	1	2	1	0	1
F	3	2	1	2	1	0

SA for PLP

- **Material Handling Cost=distance*flow**

Flow Matrix

From \ To	A	B	C	D	E	F
A	0	50	100	0	0	0
B	0	0	35	25	100	90
C	0	75	0	60	75	25
D	0	50	50	0	100	25
E	0	75	50	140	0	10
F	0	0	0	0	0	0

DistanceMatrix

	A	B	C	D	E	F
A	0	1	2	1	2	3
B	1	0	1	2	1	2
C	2	1	0	3	2	1
D	1	2	3	0	1	2
E	2	1	2	1	0	1
F	3	2	1	2	1	0

MHC

	A	B	C	D	E	F
A	0	50	200	0	0	0
B	0	0	35	50	100	180
C	0	75	0	180	150	25
D	0	100	150	0	100	50
E	0	75	100	140	0	10
F	0	0	0	0	0	0

SA for PLP

- **Initial Solution**

1	2	3	4	5	6
A	B	C	D	E	F

MHC

	A	B	C	D	E	F	Subtotal
A	0	50	200	0	0	0	250
B	0	0	35	50	100	180	365
C	0	75	0	180	150	25	430
D	0	100	150	0	100	50	400
E	0	75	100	140	0	10	325
F	0	0	0	0	0	0	0

Total Material Handling Cost (TMHC)=1770

SA for PLP

- **Neighboring Solution**

Swapping can be used as a neighborhood operator to generate feasible neighboring solution

Current:

1	2	3	4	5	6
A	B	C	D	E	F

New:

1	2	3	4	5	6
A	D	C	B	E	F

SA for PLP

- **Annealing Schedule**
  - ◊ **Initial temperature (T_o)=100**
  - ◊ **Final temperature (T_f)=0.01**
  - ◊ **Temperature Decay $\Rightarrow$ Geometric $T=T_o\alpha^i$**

    where $\alpha=0.85$, *i* is iteration index.
  - ◊ **Iteration at each temperature=2**

SA for PLP

- **Iteration-1** (T_0=100)

Initial Solution:

1	2	3	4	5	6
A	B	C	D	E	F

Cost=1770

Neighboring Solution:

1	2	3	4	5	6
A	D	C	B	E	F

A 1	D 2	C 3
B 4	E 5	F 6

Distance Matrix

	A	B	C	D	E	F
A	0	1	2	1	2	3
B	1	0	3	2	1	2
C	2	3	0	1	2	1
D	1	2	1	0	1	2
E	2	1	2	1	0	1
F	3	2	1	2	1	0

SA for PLP

- **Iteration-1** (T_0=100)

Neighboring Solution:

1	2	3	4	5	6
A	D	C	B	E	F

MHC

	A	B	C	D	E	F	Subtotal
A	0	50	200	0	0	0	250
B	0	0	105	50	100	180	435
C	0	225	0	60	150	25	460
D	0	100	50	0	100	50	300
E	0	75	50	280	0	10	415
F	0	0	0	0	0	0	0

Total Material Handling Cost (TMHC)=1860

Since the cost is not improved, we have to use transition probability.

SA for PLP

- **Transition Probability**

$$P = e^{-\Delta f / T} = e^{-(1860-1770)/100} = 0.407$$

Assume randomly generated number r=0.2

Since **p>r** then accept the non-improving solution…

SA for PLP

- **Iteration-2**

Current Solution:

1	2	3	4	5	6
A	D	C	B	E	F

Cost=1860

Neighboring Solution:

1	2	3	4	5	6
D	A	C	B	E	F

D 1	A 2	C 3
B 4	E 5	F 6

	A	B	C	D	E	F
A	0	2	2	1	1	2
B	2	0	3	1	1	2
C	1	3	0	2	2	1
D	1	1	2	0	2	3
E	1	1	2	2	0	1
F	2	2	1	3	1	0

Distance Matrix

SA for PLP

- **Iteration-2**

Neighboring Solution:

1	2	3	4	5	6
D	A	C	B	E	F

MHC

	A	B	C	D	E	F	Subtotal
A	0	100	100	0	0	0	200
B	0	0	105	25	100	180	410
C	0	225	0	120	150	25	520
D	0	50	100	0	200	75	425
E	0	75	100	280	0	10	465
F	0	0	0	0	0	0	0

Total Material Handling Cost (TMHC)=2020

Since the cost is not improved, we have to use transition probability.

SA for PLP

- **Iteration-2**

$$P = e^{-\Delta f / T} = e^{-(2020-1860)/100} = 0.202$$

Assume randomly generated number r=0.6

Since **p<r** then reject the non-improving solution…

SA for PLP

- **Iteration-2**

Best solution so far is:

1	2	3	4	5	6
A	D	C	B	E	F

Cost=1860

Temperature update after 2 iterations:

$$T = T_o \alpha^i = 100(0.85)^1 = 85^o$$

References

1. N. Metropolis, A. W. Rosenbluth, M. Rosenbluth, A. H. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines", J. Chem. Phys.21, pp. 1087-1092, 1953.
2. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing", Science 220, 671-680, 1983.
3. M. Kamel. ECE493T8: Cooperative and Adaptive Algorithms. University of Waterloo, 2009-2010.
4. Premchand Akella. Simulated Annealing, The UMass Amherst College of Engineering, 2004.
5. Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. 2nd Edition, John Wiley & Sons Ltd, 2007.
6. G. Pataki, "Teaching integer programming formulations using the traveling salesman problem", Society for Industrial and Applied Mathematics, 45 (1), 116-123. 2003.