*Faculty of Engineering & Materials Science (EMS)*
*Mechatronics Engineering Department*

# Optimization Techniques of Multi-Cooperative Systems (MCTR 1021)

## Tutorial (03)

# Simulated Annealing (SA)

Presented by:

**Mohamed A. Ibrahim, Jessica Magdy, Dalia Mamdouh, Mohamed Ashraf, Abdulrahman Yasser**

# Outline

- Tutorial 1 Recap
- Metaheuristics
- Simulated Annealing
- Application to Function Optimization
- Case Study: Assembly Line Balancing Problem

# Tutorial 1 Recap:

## *Optimization*

**Optimization Problem** + **Optimization Algorithm** = **Optimization Solution**

### Objective Function

- Minimize travelling time
- Minimize total distance
- Minimize consumed fuel
- Maximize #ppl in car
- Minimize traffic congestion

### Decision Variables

- Time from point to point
- Distance from p2p
- Fuel amount in car
- Vehicle capacity
- Traffic status

### Constraints

- Vehicle Capacity
- Money for fuel
- Need of ppl
- Be in uni before 1st slot

- **Tutorial 1 Recap**
- Metaheuristics
- Simulated Annealing
- Application to Function Optimization
- Assembly Line Balancing Problem

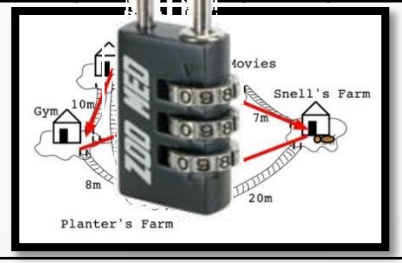
# Tutorial 1 Recap:

## *Optimization*

**Optimization Problem** + **Optimization Algorithm** = **Optimization Solution**

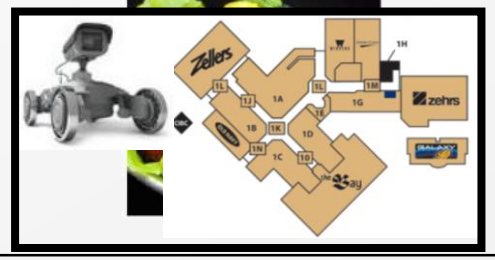
### Permutation

| A | C | | B | E |
|---|---|---|---|---|




### Combination

| (R1,T2) | (R2,T1) | (R3,T3) |
|---|---|---|



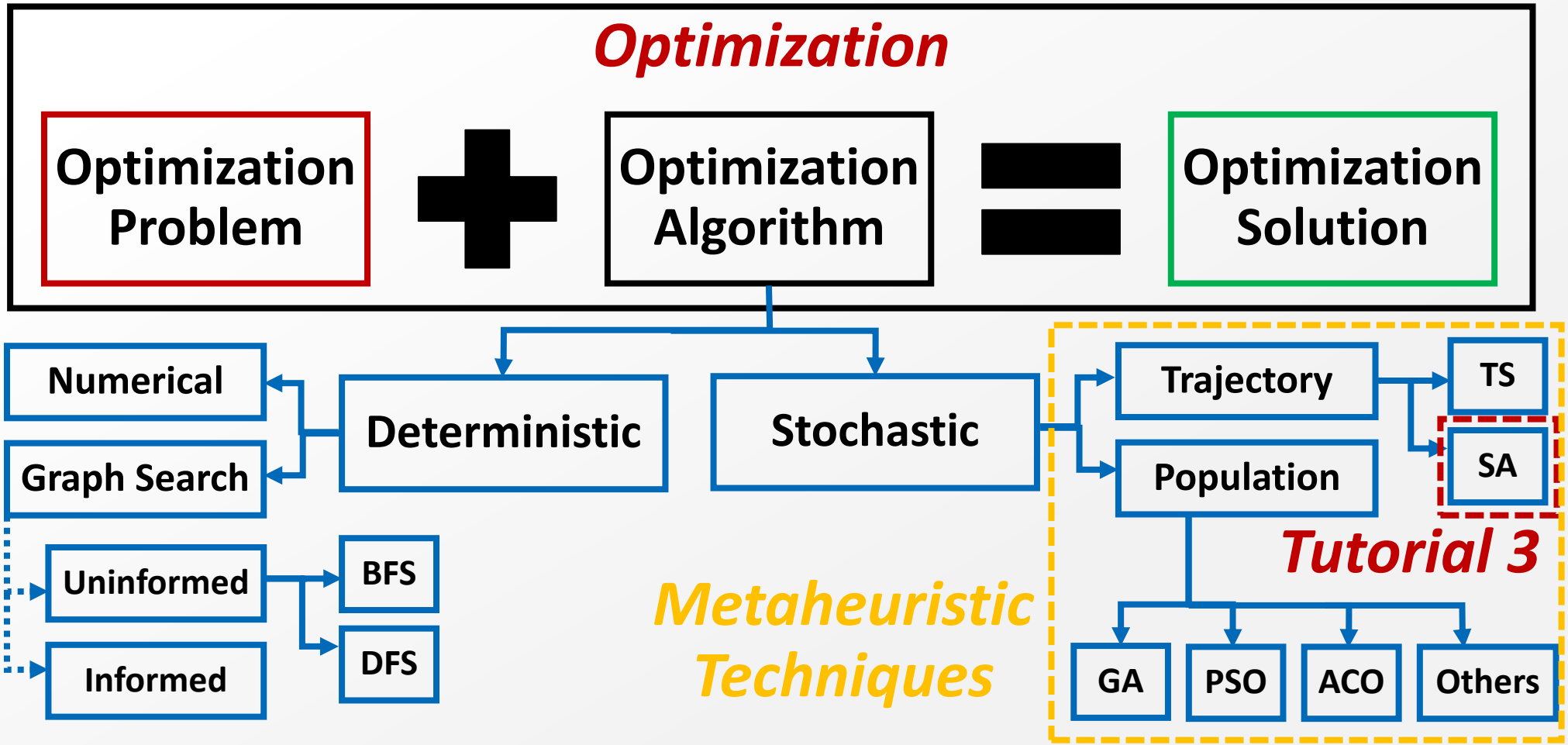

### Binary

| 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|

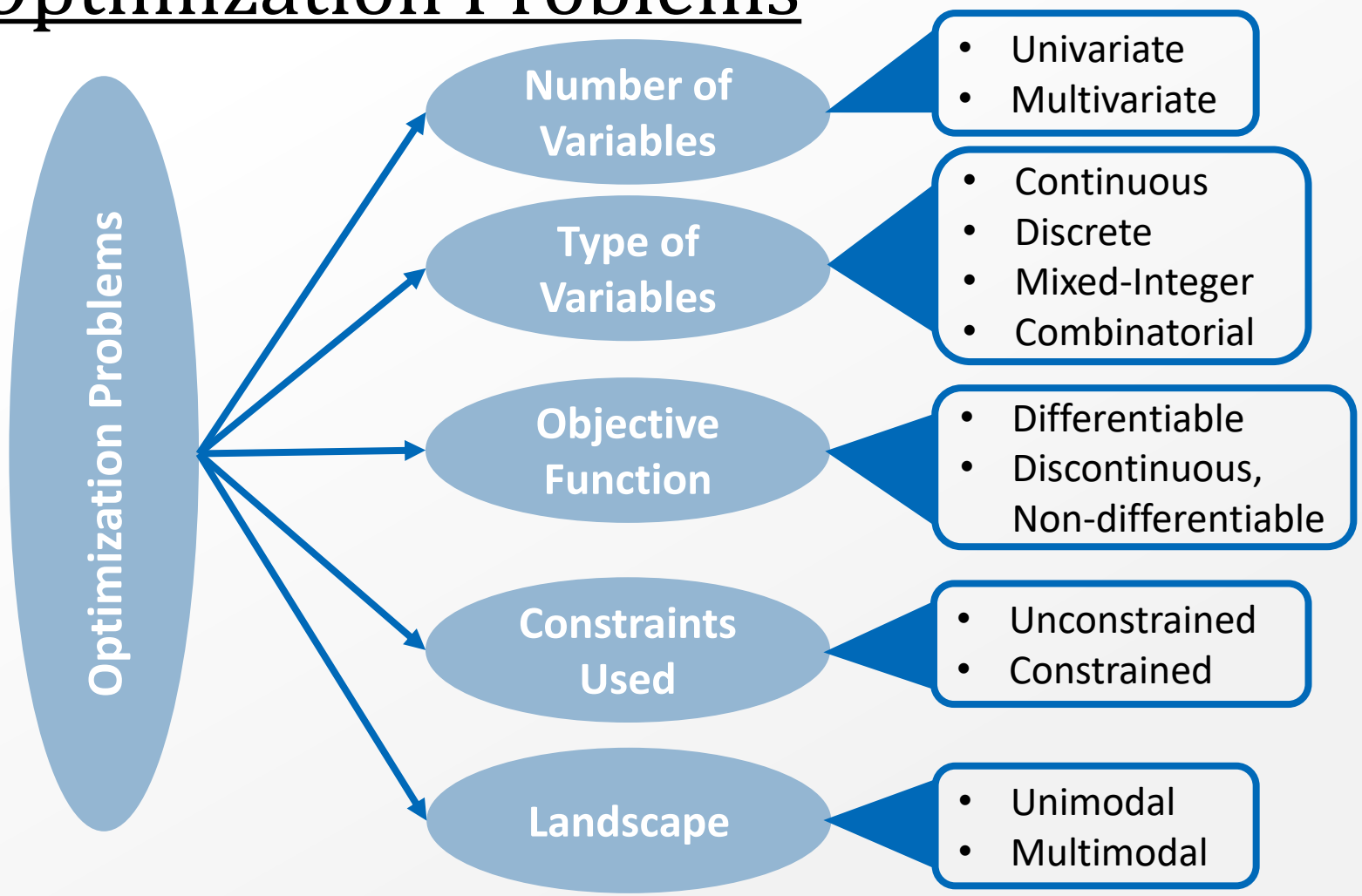

### Arithmetic

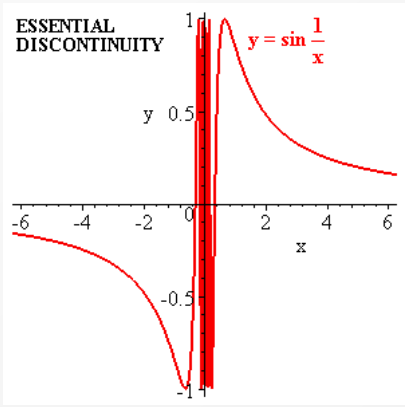
| 31 | 23 | 19 | 84 | 21 |
|---|---|---|---|---|

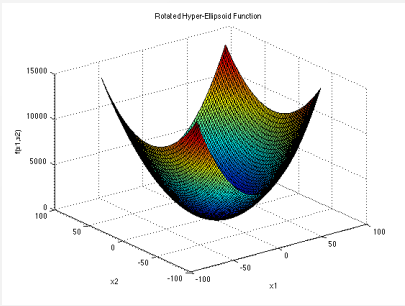
# Tutorial 1 Recap:

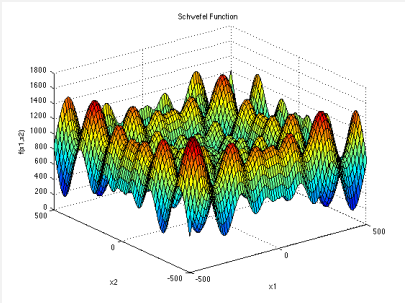


# Optimization Problems



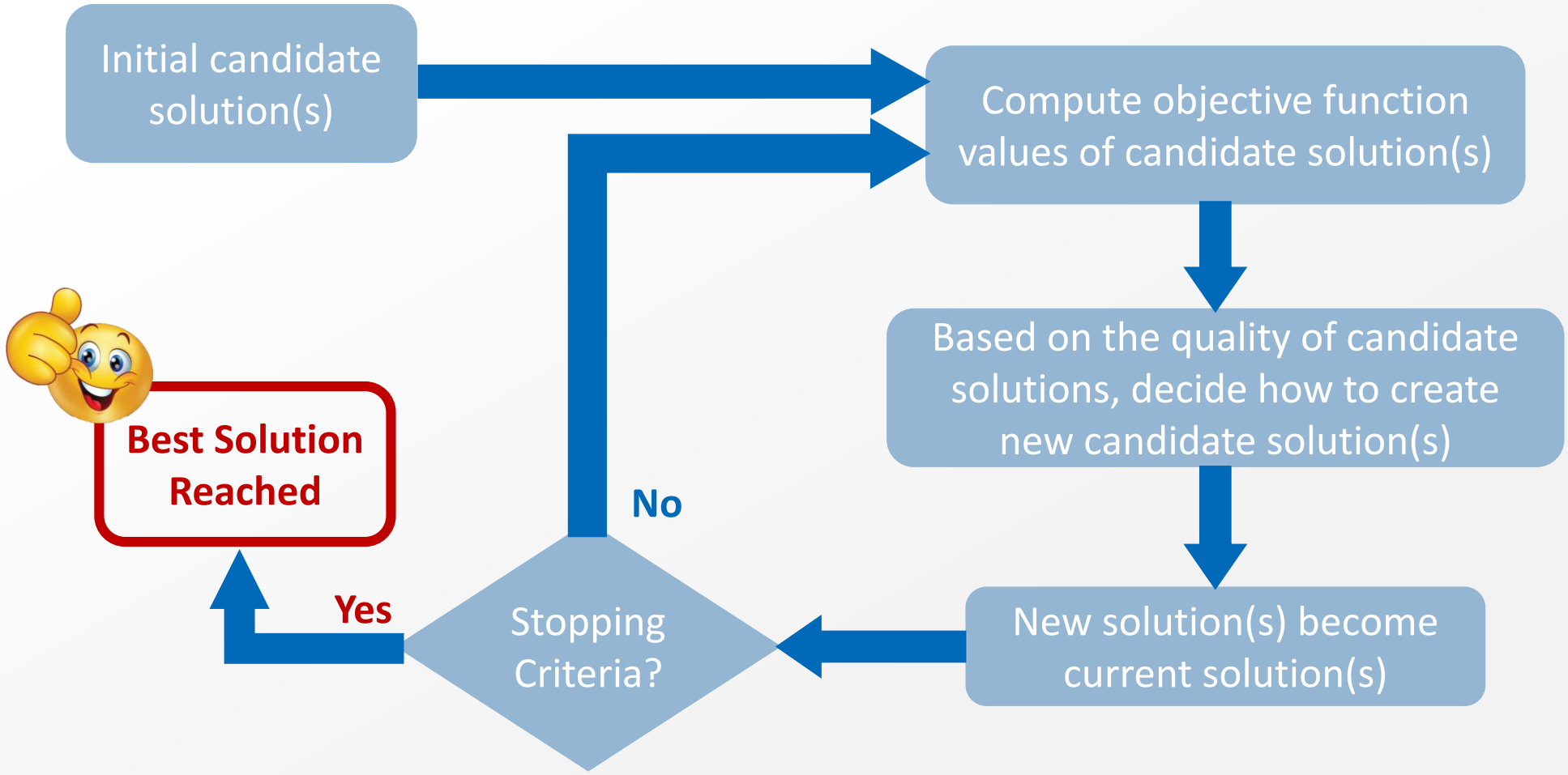



# What is a Metaheuristic?

- ***What is the situation?***
  - Extremely large solution space
  - Lack of algorithmic solutions (i.e. we don't know exactly what good features the best solutions have)
  - Non-differentiable mathematical model
- ***What can we do?***
  - We know the solution representation and the feasibility conditions
  - We can generate an initial random feasible solution
  - We can modify this solution to create a new solution instance
  - We can do this repetitively to improve our solution quality
- ***This is the idea behind all metaheuristics.***

- Tutorial 1 Recap
- **Metaheuristics**
- Simulated Annealing
- Application to Function Optimization
- Assembly Line Balancing Problem

# What is a Metaheuristic?



# What is a Metaheuristic?

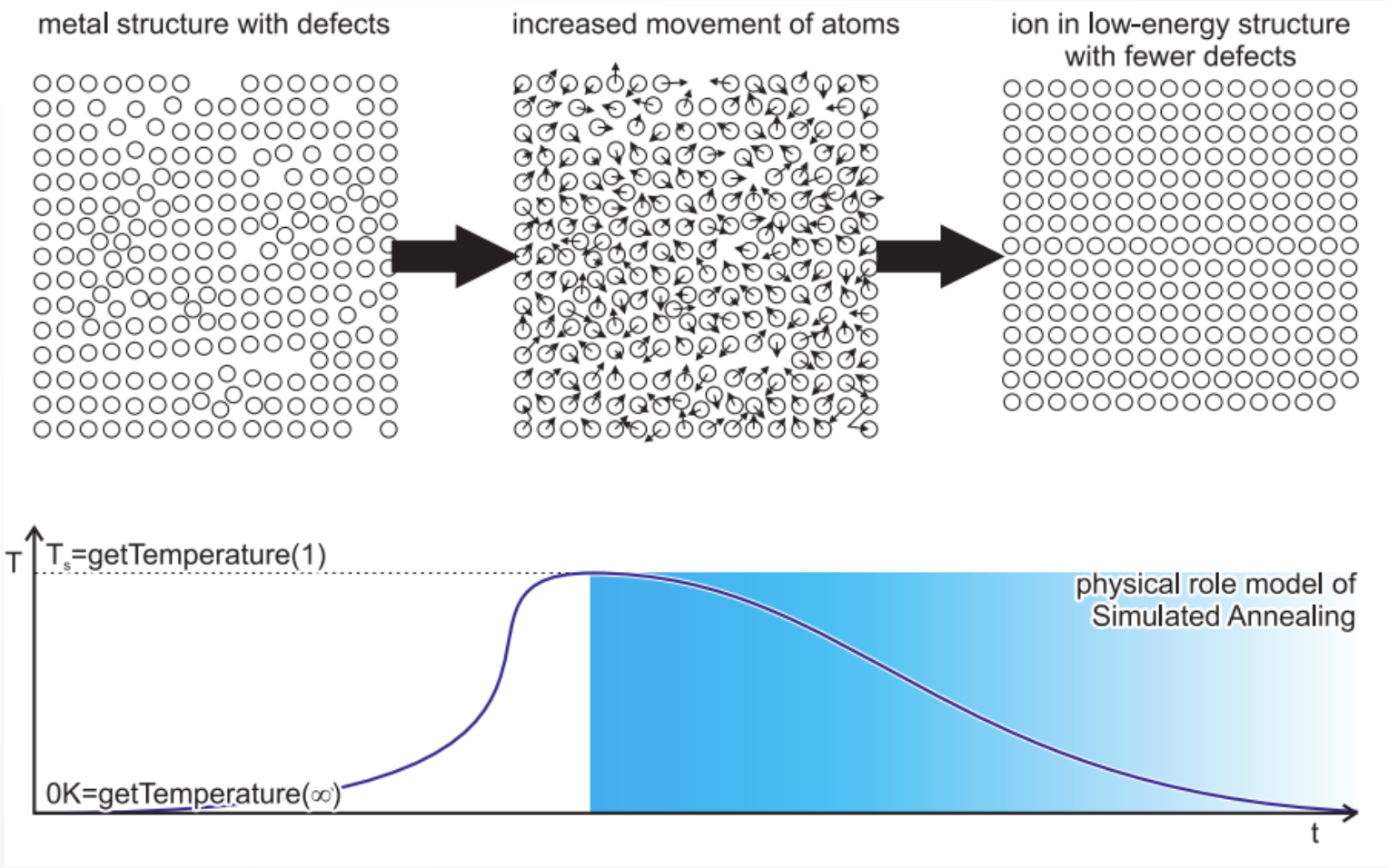
- It is a ***higher-level procedure*** to provide a ***near-optimal solution*** (i.e. sufficiently good) to an optimization problem – especially with incomplete/imperfect information or limited computational capacity.
- Metaheuristics ***randomly sample*** a subset of solutions which are otherwise too large to be explored.
- So, metaheuristics are like a *black box* that can be implemented on many problems to provide *good* solutions.
- Often, we need to add *domain-specific knowledge* to these metaheuristic algorithms to obtain *excellent* solutions.

# Physical Annealing

- ***Physical Annealing*** is a metal heat treatment that alters the microstructure of the material causing changes in properties (e.g. strength, hardness, ductility, … etc)
- Physical Annealing Process:
  - Heat the metal beyond the recrystallization temperature for a suitable time.
  - Metal is cooled down slowly under a controlled rate.
- As the metal ***cools down slowly***, the ions assume ***minimum-energy*** stable positions in the crystal. So, an initial brittle structure is transformed to a ***better and more stable configuration*** (e.g. refined grains, eliminate defects and residual stresses, improve ductility … etc)

- Tutorial 1 Recap
- Metaheuristics
- **Simulated Annealing**
- Application to Function Optimization
- Assembly Line Balancing Problem

# Physical Annealing



# Simulated Annealing

- ***Simulated Annealing (SA)*** is a meta-heuristic optimization algorithm emulating the physical annealing process.
- Metropolis et al. (1953) proposed a Monte Carlo algorithm to simulate the physical annealing process. Then, Kirkpatrick (1982) applied SA to optimization problems to find near-optimal solutions in large solution spaces.'
- In the context of optimization, the ***minimum of the objective function*** represents the ***minimum-energy state of the system***.

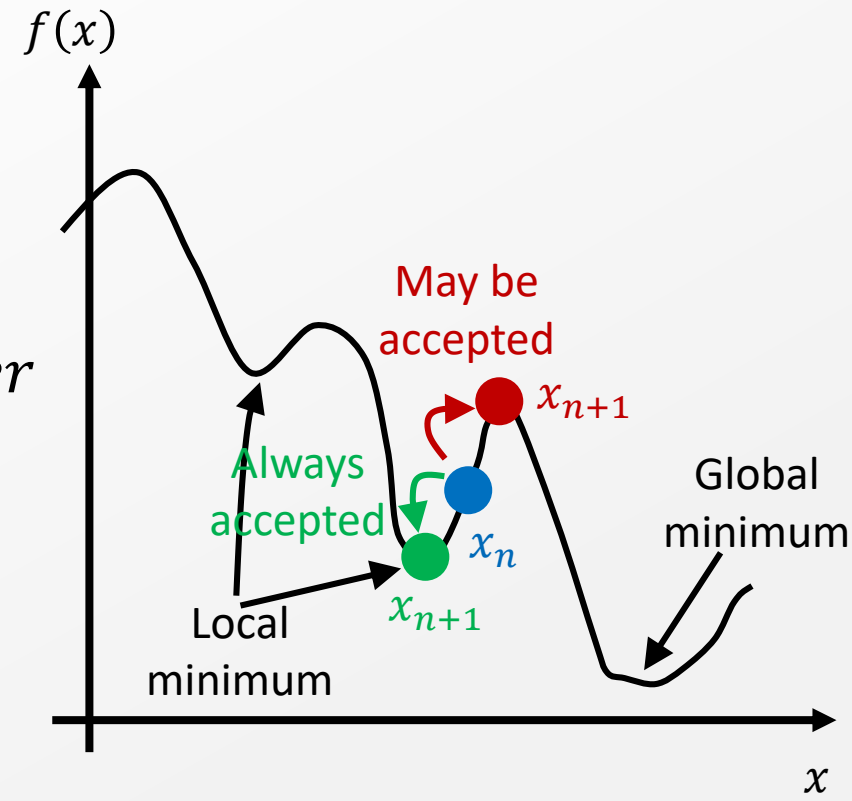
# Simulated Annealing

| Optimization Algorithm | Simulated Annealing Algorithm |
|---|---|
| Decision Variable | Position of molecule in the substance |
| Solution<br>(i.e. vector of all decision variables) | State of the substance |
| Objective Function | Energy of the substance state |
| Generation of New Solutions | Random movement of molecules |
| Selection of Solutions | • If new solution has a lower energy state, then it is always accepted.<br>• Otherwise, solution is accepted probabilistically. |

# Simulated Annealing

- ***Transition Probability*** is the probability of moving from the current solution $x_i$ to a new solution $x_j$.

$$p_{ij} = \begin{cases} 1; if\ f(x_j) < f(x_i)\ \ldots \sim x_j\ is\ better \\ e^{\frac{-\Delta E}{T}}; otherwise \end{cases}$$




- This means that you **always** go to **better solutions**.

- However, **worse solutions** are **accepted probabilistically** based on the temperature. For high temperature initially, worse solutions are accepted more to explore search space. Also, this helps to get out of being stuck of local minima.

# Simulated Annealing

- ***Linear Cooling Schedule:***

$$T_{cur} = T_0 - \beta i_{cur}$$
$$where\ \beta = \frac{T_f - T_0}{i_{max}}$$

*Where $\beta$ is the cooling rate and $i_{cur}$ is the current iteration number .*

- ***Geometric Cooling Schedule:***

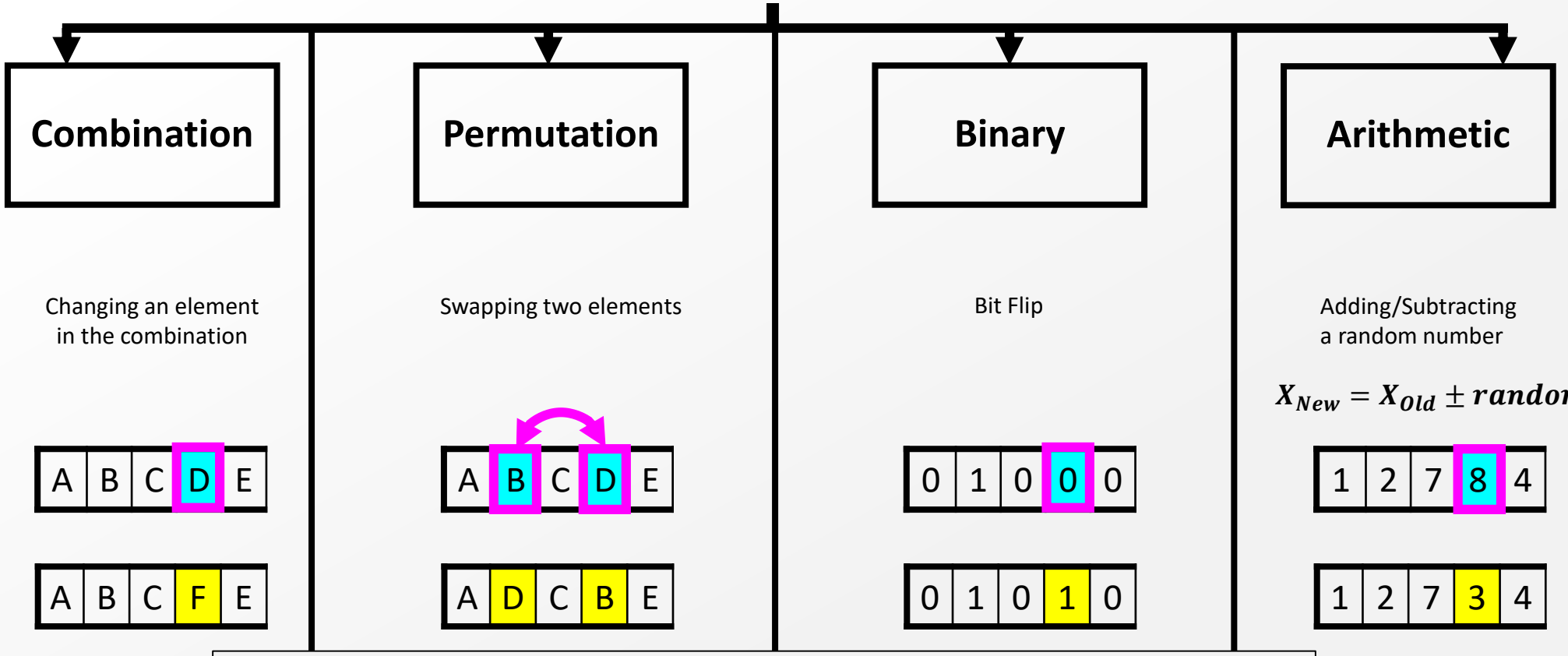
$$T_{cur} = T_0 \alpha^{i_{cur}}$$
$$where\ 0 < \alpha < 1\ ;\alpha = 0.7 \sim 0.95$$

*Where $\alpha$ is the cooling rate and $i_{cur}$ is the current iteration number.*

- You can have multiple iterations $n_T$ per a single temperature.

# Simulated Annealing

***Random Movement****(Examples)*




Examples to possible random movements. There are more examples in the literature

- Tutorial 1 Recap
- Metaheuristics
- **Simulated Annealing**
- Application to Function Optimization
- Assembly Line Balancing Problem

# Simulated Annealing

**Initialization:**
- $x_0$ (initial feasible solution)
- $T_0$ (initial temperature)
- $T_f$ (final temperature)
- $i_{max}$ (maximum number of iterations)

**Generate New Random Solutions:**
$x_{new} = x_{cur} + \delta$
$\delta = random\ movement$

**Compute change in energy state:**
$\Delta E = \Delta f = f(x_{new}) - f(x_{cur})$

**Solution better?**
$\Delta E < 0?$

Yes → **Accept better solution:** $x_{cur} = x_{new}$

No → $p = e^{\frac{-\Delta E}{T}} > r?$

Yes → **Accept worse solution:** $x_{cur} = x_{new}$

No

**Update best solution reached:**
$if\ f(x_{cur}) < f^*: x^*, f^* = x_{cur}, f(x_{cur})$

**Update temperature according to cooling schedule.**

**Continue?**
$T_{cur} > T_f,$
$i < i_{max}?$

Yes → Generate New Random Solutions

No → **Best Solution Reached**

# Simulated Annealing Algorithm

- Decision Variables: $\underline{\boldsymbol{x}} = x_1, x_2, \dots, x_p$
- Objective Function: $f(\underline{\boldsymbol{x}})$

Initialize $\underline{\boldsymbol{x}}_0, T_0, T_f, i_{max}, x^*, f^*$
While $T_{cur} > T_f$ and $i \leq i_{max}$:
    Repeat for $n_T$ (iterations per temperature):
        Generate random new **feasible** solutions: $x_{new} = x_{cur} + rand$
        Compute change in energy: $\Delta E = \Delta f = f(x_{new}) - f(x_{cur})$
        If $\Delta E < 0$:
            Accept better solution: $f(x_{new})$
        else if $r < p = e^{\frac{-\Delta E}{T}}$ ( r is a random number 0~1):
            Accept solution: $x_{cur} = x_{new}$
        If $f(x_{cur}) < f^*$:
            Update best reached solution: $x^*, f^* = x_{cur}, f(x_{cur})$
    Update temperature according to cooling schedule.
    Update iteration counter: $i = i + 1$
Output best reached solution $x^*, f^*$

# Function Minimization using SA

- ***Problem Statement:***
  Find the minimum point of the function:

$$f(x) = -x^3 + 60x^2 - 900x - 100$$
$$subject\ to: 0 \leq x \leq 32$$

- Use SA with the following parameters:
  - $n_T = 2$ iterations per temperature
  - $T_0 = 1000$ (initial temperature)
  - $T_f = 0.01$ (final temperature)
  - $i_{max} = 200$ (maximum number of iterations)
  - Linear cooling schedule
  - Assume initial guess: $x_0 = 2$
- Perform 2 iterations only.

# Function Minimization using SA

- ***Initialization:***
  - Initial random **feasible** solution: $x_0 = 2$
  - Initial objective function value:
    $$f(x_0) = -2^3 + 60(2^2) - 900(2) - 100 = -1668$$
  - Best Solution so far: $x^* = 2$ with $f^* = -1668$
  - Initial temperature: $T = 1000$
  - Linear cooling rate: $\beta = \frac{T_f - T_0}{i_{max}} = \frac{1000 - 0.01}{200} \approx 5$

# Function Minimization using SA

- ***Iteration #1a:***
  - Perform random movement:
    $$x_1 = x_0 + rand(0{\sim}32) = 8.5$$
  - Compute change in energy:
    $$f(x_1) = -x^3 + 60x^2 - 900x - 100 = -4029$$
    $$\Delta E = \Delta f = f(x_1) - f(x_0) = -4029 - (-1668) = -2361$$
  - This is a better solution $\Delta E < 0$; so, it is accepted.
    $$x_{cur} = x_1 = 8.5$$
  - Update best solution so far (since $f(x_1) < f^*$):
    $$x^* = 8.5 \; with \; f^* = -4029$$
  - This will be repeated again as we perform $n_T = 2$ iterations per temperature.

# Function Minimization using SA

- ***Iteration #1b:***
  - Perform random movement:
$$x_2 = x_{cur} + rand(0{\sim}32) = 12.8$$
  - Compute change in energy:
$$f(x_1) = -x^3 + 60x^2 - 900x - 100 = -3887$$
$$\Delta E = \Delta f = f(x_1) - f(x_0) = -3887 - (-4029) = 142$$
  - This is a worse solution $\Delta E > 0$; so, we calculate the transition probability.
$$p = e^{\frac{-\Delta E}{T}} = e^{\frac{-142}{1000}} = 0.868$$
  - Generate a random number: $r = 0.4$
  - Since $r < p$, then it is accepted:
$$x_{cur} = x_2 = 12.8$$
  - Best solution so far is not updated.
  - Update temperature:
$$T = T_0 - 5i_{cur} = 1000 - 5 = 995$$

# Function Minimization using SA

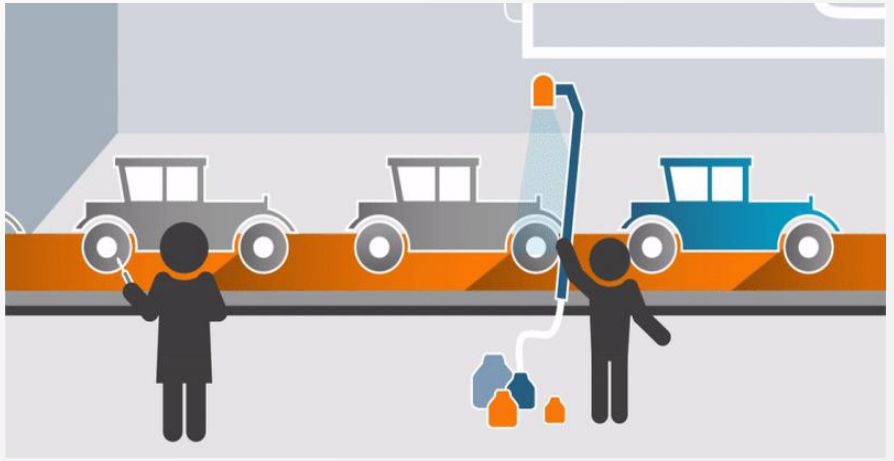
- ***Iteration #2a:***
  - Perform random movement:
$$x_3 = x_{cur} + rand(0{\sim}32) = 15$$
  - Compute change in energy:
$$f(x_1) = -x^3 + 60x^2 - 900x - 100 = -3475$$
$$\Delta E = \Delta f = f(x_1) - f(x_0) = -3475 - (-3887) = 412$$
  - This is a worse solution $\Delta E > 0$; so, we calculate the transition probability.
$$p = e^{\frac{-\Delta E}{T}} = e^{\frac{-412}{995}} = 0.661$$
  - Generate a random number: $r = 0.8$
  - Since $r > p$, then it is NOT accepted:
$$x_{cur} = x_2 = 12.8$$
  - Best solution so far is not updated.
  - This will be repeated again as we perform $n_T = 2$ iterations per temperature.

# Function Minimization using SA

- ***Iteration #2b:***
  - Perform random movement:
    $$x_4 = x_{cur} + rand(0{\sim}32) = 10$$
  - Compute change in energy:
    $$f(x_1) = -x^3 + 60x^2 - 900x - 100 = -4100$$
    $$\Delta E = \Delta f = f(x_1) - f(x_0) = -4100 - (-3887) = -213$$
  - This is a better solution $\Delta E < 0$; so, it is accepted.
    $$x_{cur} = x_4 = 10$$
  - Update best solution so far (since $f(x_1) < f^*$):
    $$x^* = 10 \; with \; f^* = -4100$$
  - Update temperature:
    $$T = T_0 - 5i_{cur} = 1000 - 5(2) = 990$$
- And the iterations continue ...

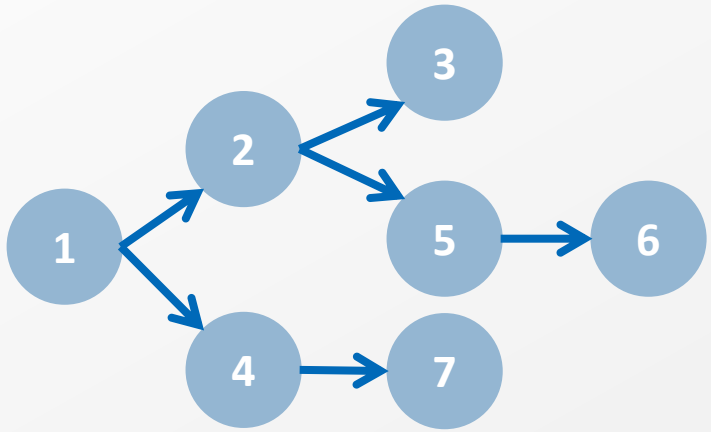
# Case Study #3: Assembly Line Balancing

- Let there be $n$ tasks (each with time requirement $t_i$) to be completed by $m$ workstations with a predefined sequence.
- **Decision Variables:**
  - Which task $i$ is done by which workstation $j$.
- **Objective Function:**
  - Minimize the number of used workstations.
- **Constraint Equation and Inequalities:**
  - Sequential order of tasks (i.e. precedence constraints).
  - Cycle time of the assembly line.
  - Maximum number of workstations.



# Assembly Line Balancing: Formulation

- **Given Data:**
  - Time Requirements of each task:
    $$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$$
  - Required cycle time: $C = 8$
  - Precedence Constraints:
    - This can be represented as a vector of tuples.
      $P = [(1,2), (1,4), (2,3), (2,5), (4,7), (5,6)]$
    - Or it can be represented as a precedence matrix.

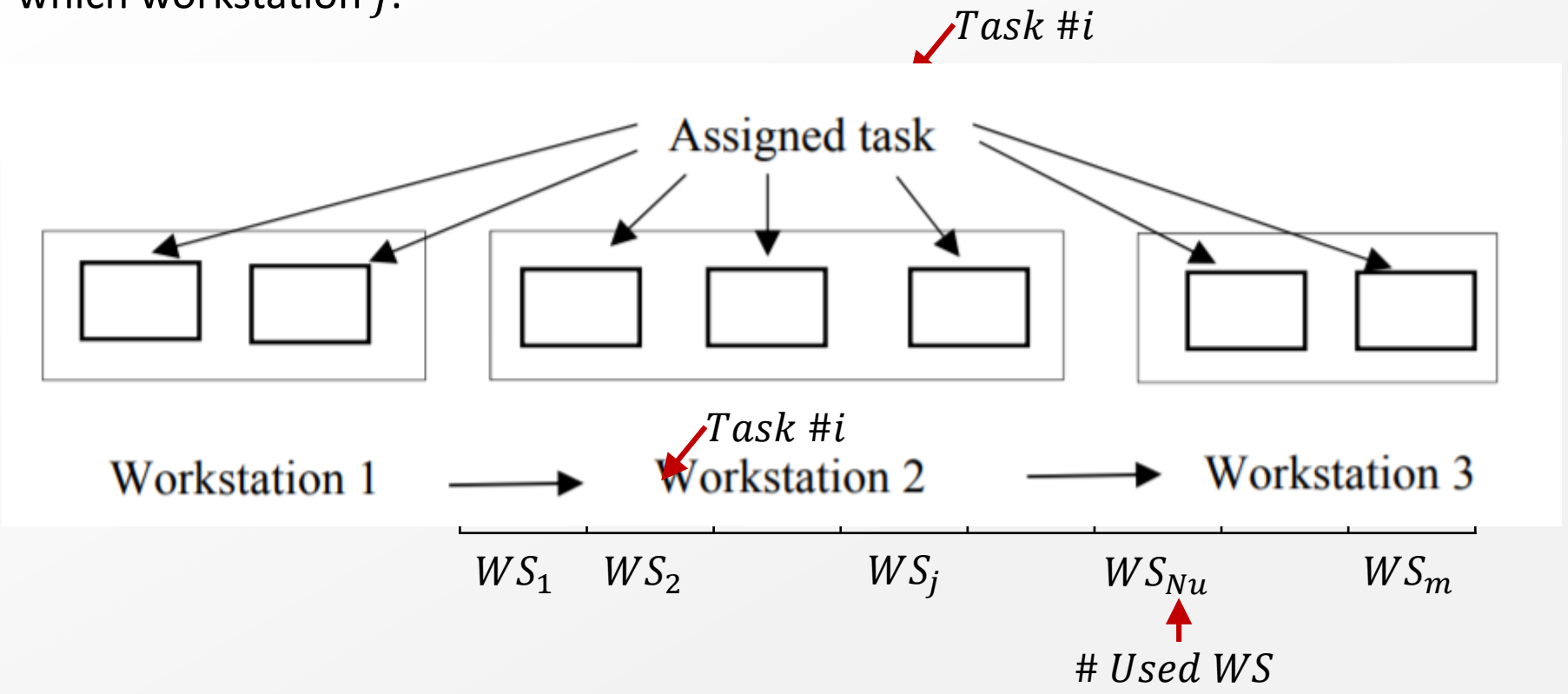



$P_{ij}$ is whether task $i$ has a precedence relation with task $j$.

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T2 | | 0 | 1 | 0 | 1 | 0 | 0 |
| T3 | | | 0 | 0 | 0 | 0 | 0 |
| T4 | | | | 0 | 0 | 0 | 1 |
| T5 | | | | | 0 | 1 | 0 |
| T6 | | | | | | 0 | 0 |
| T7 | | | | | | | 0 |

# Assembly Line Balancing: Formulation

- **Solution Representation:**
  - Which task $i$ is done by which workstation $j$.



# Assembly Line Balancing: Formulation

- **Solution Representation:**
  - Which task $i$ is done by which workstation $j$.

$N_u = 5$

| | | | | | | |
|---|---|---|---|---|---|---|
| $Solution\ 1 \rightarrow$ | 1 | 2,4 | 3 | 5 | 6,7 | - |
| | $WS_1$ | $WS_2$ | | $WS_j$ | | $WS_6$ |
| $Solution\ 2 \rightarrow$ | 1 | 2 | 3,4 | 6 | 5 | 7 |

$N_u = 6$

- **Objective Function:**
  - Minimize the number of used workstations.

$$\min f = \min N_u$$

  - OR we can minimize the smoothing index (to make sure the workstations are equally loaded as well).

$$\min f = \min SI$$

$$SI = \sqrt{\frac{\sum_{i=1}^{N_u}(WL_i - WL_{max})^2}{N_u}}$$

$N_u = number\ of\ used\ WS$
$WL_i = work\ load\ of\ WS_i$
$WL_{max} = maximum\ workload\ across\ all\ WS$

# Assembly Line Balancing: Formulation

- **Solution Representation:**
  - Which task $i$ is done by which workstation $j$.

$Solution\ 1 \rightarrow$

| 1 | ~~2~~,4 | ~~3~~ | 5 | ~~6~~,7 | ~~7~~ |
|---|---|---|---|---|---|
| $WS_1$ | $WS_2$ | | $WS_j$ | | $WS_6$ |

$N_u = 6$

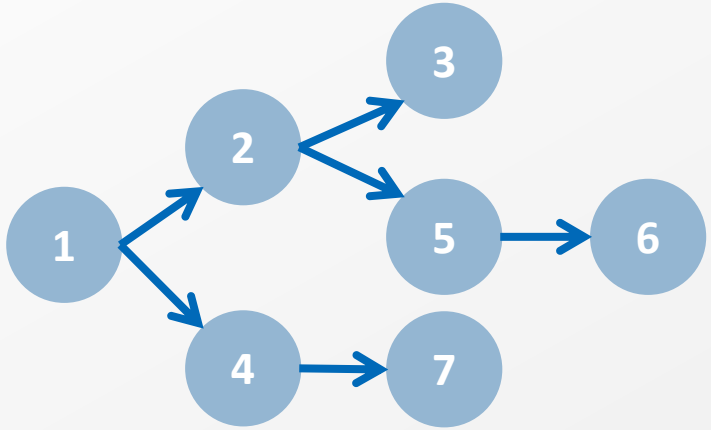
Time Requirements of each task:
$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$

- **Constraint Equation and Inequalities:**
  - Sequential order of tasks (i.e. precedence constraints).
  - Cycle time of the assembly line.
    $$WL_i \leq C = 8$$
    $$WL_i = \sum t_j \; ; if \; T_j \; in \; WS_i$$
  - Maximum number of workstations.
    $$N_u \leq m = 6$$

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T2 | | 0 | 1 | 0 | 1 | 0 | 0 |
| T3 | | | 0 | 0 | 0 | 0 | 0 |
| T4 | | | | 0 | 0 | 0 | 1 |
| T5 | | | | | 0 | 1 | 0 |
| T6 | | | | | | 0 | 0 |
| T7 | | | | | | | 0 |

# Assembly Line Balancing: SA

- **Given Data:**
  - Time Requirements of each task:
    $$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$$
  - Required cycle time: $C = 8$
  - Maximum number of workstations: $m = 6$
  - Optimize the smoothing index of the AL.




- Use SA with the following parameters:
  - $n_T = 1$ iteration per temperature
  - $T_0 = 100$ (initial temperature)
  - $T_f = 0.01$ (final temperature)
  - $i_{max} = 200$ (maximum number of iterations)
  - Geometric cooling schedule with $\alpha = 0.8$
  - Perform 2 iterations only.

# Assembly Line Balancing: SA

Cycle Time: $C = 8$
Max. workstations: $m = 6$
Time Requirements of each task:
$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$

- ***Initialization:***
  - Initial random **feasible** solution:

| $l_0 \rightarrow$ | 1 | 2 | 3,4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| | $WS_1$ | $WS_2$ | | | | $WS_6$ |
| $WL_i \rightarrow$ | 1 | 5 | 7 | 5 | 6 | 5 |

  - Initial objective function value:

$$SI(l_0) = \sqrt{\frac{\sum_{i=1}^{N_u}(WL_i - WL_{max})^2}{N_u}} = 2.86$$

  - Best Solution so far: $x^* = l_0$ with $SI^* = 2.86$
  - Initial temperature: $T = 100$

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T2 | | 0 | 1 | 0 | 1 | 0 | 0 |
| T3 | | | 0 | 0 | 0 | 0 | 0 |
| T4 | | | | 0 | 0 | 0 | 1 |
| T5 | | | | | 0 | 1 | 0 |
| T6 | | | | | | 0 | 0 |
| T7 | | | | | | | 0 |

# Assembly Line Balancing: SA

- ***Iteration #1:***
  - Obtain new solution randomly (**remove and insert task**):

| | | | | | | |
|---|---|---|---|---|---|---|
| $l_0 \rightarrow$ | 1 | 2 | 3,4 | 5 | 6 | 7 |
| | $WS_1$ | $WS_2$ | | | | $WS_6$ |
| $l_1 \rightarrow$ | 1 | 2 | 3 | 4,5 | 6 | 7 |

$WL_i \rightarrow$ 1 5 4 8 6 5

  - New objective function value:

$$SI(l_1) = \sqrt{\frac{\sum_{i=1}^{N_u}(WL_i - WL_{max})^2}{N_u}} = 3.81$$

  - Compute change in energy:

$$\Delta E = \Delta SI = 3.81 - 2.86 = 0.95$$

  - This is a worse solution $\Delta E > 0$.

Cycle Time: $C = 8$
Max. workstations: $m = 6$
Time Requirements of each task:
$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T2 | | 0 | 1 | 0 | 1 | 0 | 0 |
| T3 | | | 0 | 0 | 0 | 0 | 0 |
| T4 | | | | 0 | 0 | 0 | 1 |
| T5 | | | | | 0 | 1 | 0 |
| T6 | | | | | | 0 | 0 |
| T7 | | | | | | | 0 |

# Assembly Line Balancing: SA

- ***Iteration #1 (continued):***
  - This is a worse solution $\Delta E > 0$; so, we calculate the transition probability.

$$p = e^{\frac{-\Delta E}{T}} = e^{\frac{-0.95}{100}} = 0.991$$

  - Generate a random number: $r = 0.8$
  - Since $r < p$, then it is accepted:

| $l_1 \rightarrow$ | 1 | 2 | 3 | 4,5 | 6 | 7 |
|---|---|---|---|---|---|---|
| $WL_i \rightarrow$ | 1 | 5 | 4 | 8 | 6 | 5 |

  - Best solution so far is not updated. $x^* = l_0$ with $SI^* = 2.86$
  - Update temperature:

$$T = T_0 \alpha^{i_{cur}} \; with \; \alpha = 0.8$$
$$T = 100(0.8)^1 = 80$$

Cycle Time: $C = 8$
Max. workstations: $m = 6$
Time Requirements of each task:
$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T2 | | 0 | 1 | 0 | 1 | 0 | 0 |
| T3 | | | 0 | 0 | 0 | 0 | 0 |
| T4 | | | | 0 | 0 | 0 | 1 |
| T5 | | | | | 0 | 1 | 0 |
| T6 | | | | | | 0 | 0 |
| T7 | | | | | | | 0 |

# Assembly Line Balancing: SA

- ***Iteration #2:***
  - Obtain new solution randomly (**remove and insert task**):

| | | | | | | |
|---|---|---|---|---|---|---|
| $l_1 \rightarrow$ | 1 | 2 | 3 | 4,5 | 6 | 7 |
| | $WS_1$ | $WS_2$ | | | | $WS_6$ |
| $l_2 \rightarrow$ | 1,2 | 3 | 4,5 | 6 | 7 | - |
| $WL_i \rightarrow$ | 6 | 4 | 8 | 6 | 5 | – |

  - New objective function value:

$$SI(l_2) = \sqrt{\frac{\sum_{i=1}^{N_u}(WL_i - WL_{max})^2}{N_u}} = 2.57$$

  - Compute change in energy:

$$\Delta E = \Delta SI = 2.57 - 3.81 = -1.24$$

  - This is a better solution $\Delta E < 0 \Rightarrow$ ACCEPTED!

Cycle Time: $C = 8$
Max. workstations: $m = 6$
Time Requirements of each task:
$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T2 | | 0 | 1 | 0 | 1 | 0 | 0 |
| T3 | | | 0 | 0 | 0 | 0 | 0 |
| T4 | | | | 0 | 0 | 0 | 1 |
| T5 | | | | | 0 | 1 | 0 |
| T6 | | | | | | 0 | 0 |
| T7 | | | | | | | 0 |

# Assembly Line Balancing: SA

- ***Iteration #2 (continued):***
  - Best solution so far is updated: $x^* = l_2$ with $SI^* = 2.57$

| $l_2 \rightarrow$ | 1,2 | 3 | 4,5 | 6 | 7 | - |
|---|---|---|---|---|---|---|
| $WL_i \rightarrow$ | 6 | 4 | 8 | 6 | 5 | – |

  - Update temperature:

$$T = T_0 \alpha^{i_{cur}} \; with \; \alpha = 0.8$$
$$T = 100(0.8)^2 = 64$$

- And the iterations continue ...

Cycle Time: $C = 8$
Max. workstations: $m = 6$
Time Requirements of each task:
$t = [t_i] = [1, 5, 4, 3, 5, 6, 5]$

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T2 | | 0 | 1 | 0 | 1 | 0 | 0 |
| T3 | | | 0 | 0 | 0 | 0 | 0 |
| T4 | | | | 0 | 0 | 0 | 1 |
| T5 | | | | | 0 | 1 | 0 |
| T6 | | | | | | 0 | 0 |
| T7 | | | | | | | 0 |


***See you Next time … ☺***