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

Optimization Techniques of Multi-Cooperative Systems (MCTR 1021)

Tutorial (04)

Genetic Algorithm (GA)

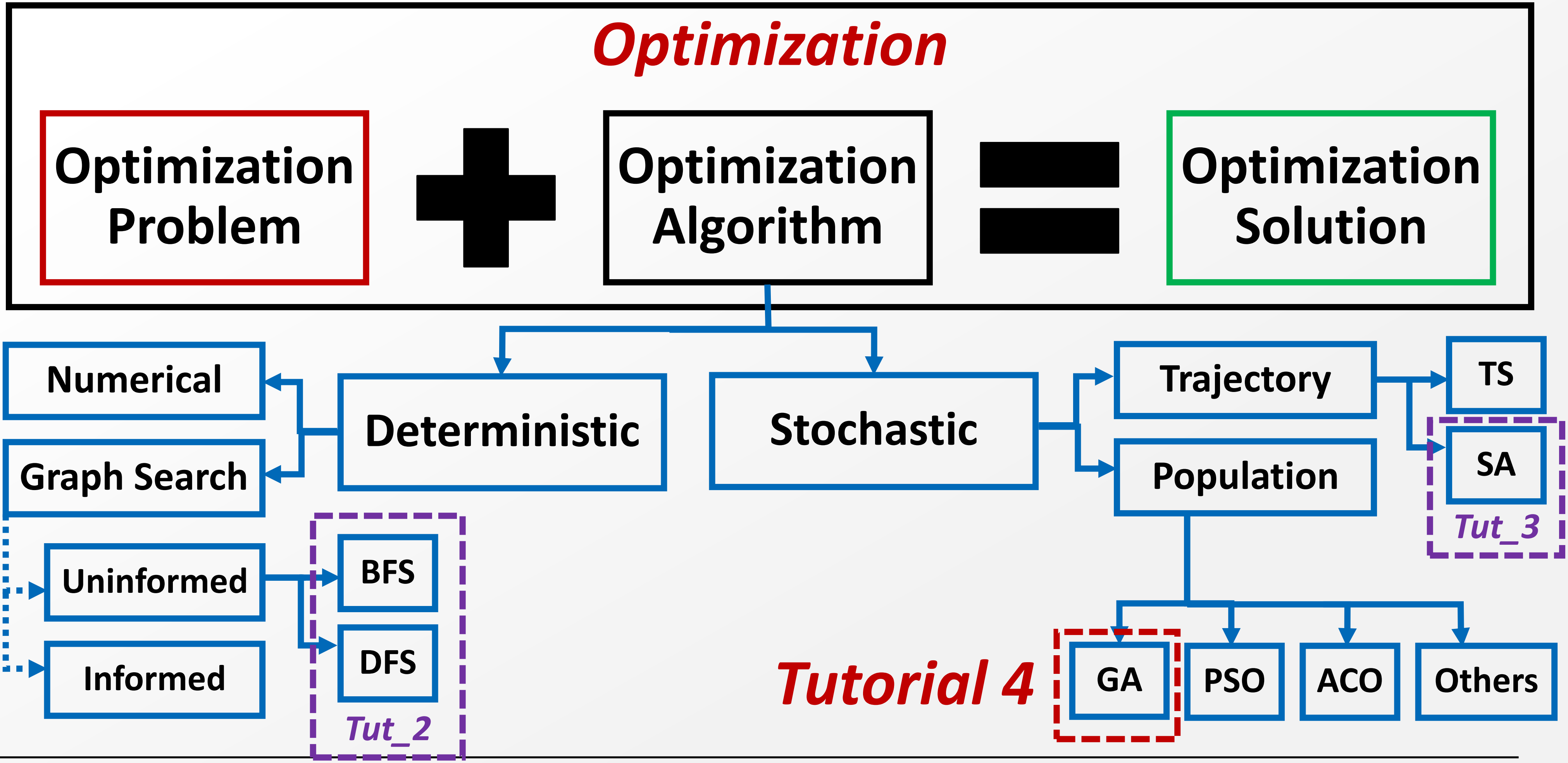
Presented by:

Mohamed A. Ibrahim, Jessica Magdy, Dalia Mamdouh, Mohamed Ashraf, Abdulrahman Yasser, Dr. Omar M. Shehata

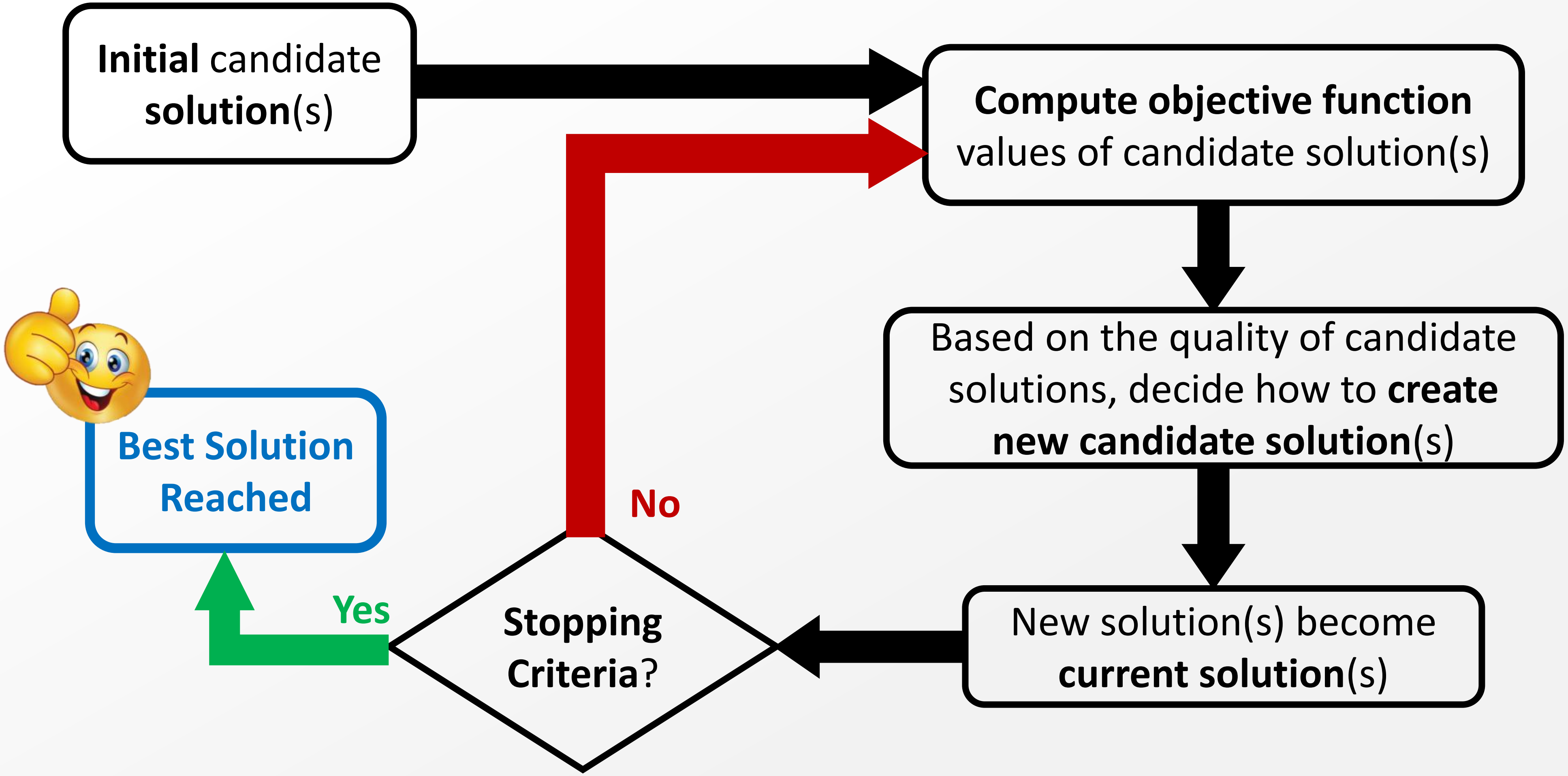
Outline

- Tutorial 3 Recap
- Genetic Algorithm
- Application to Function Optimization
- Assembly Line Balancing Problem
- Techniques Comparison

Tutorial 3 Recap: Course Overview

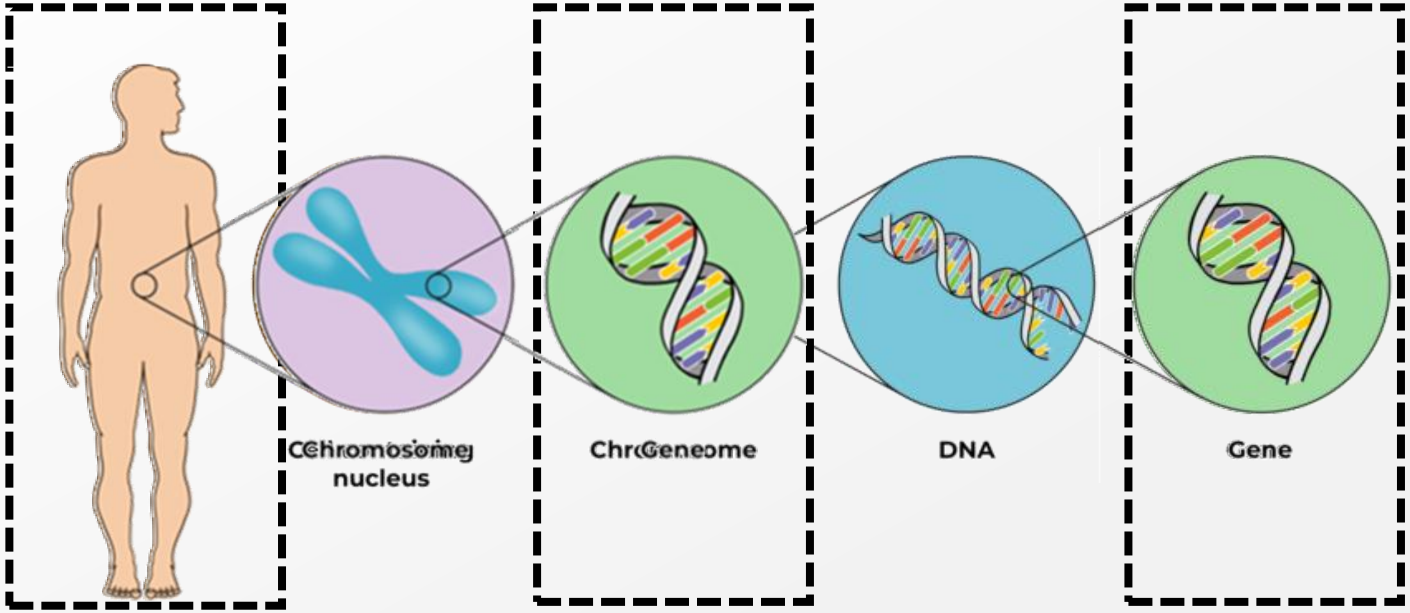


Tutorial 3 Recap: Metaheuristics Flow Chart



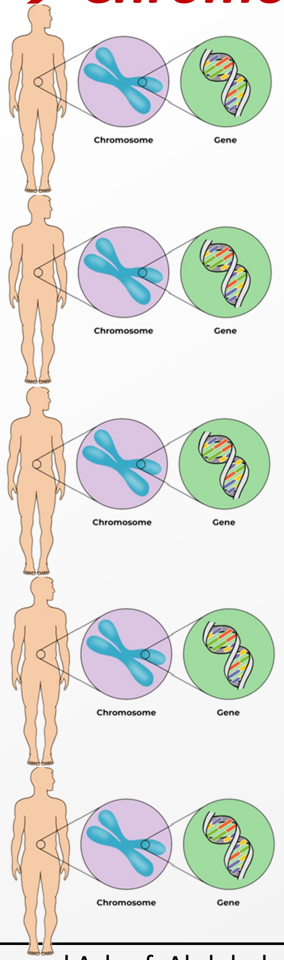
Genetic Algorithm

- Genetic Algorithm (GA) was developed by Holland (1975) and was popularized by Goldberg (1989) inspired by the **genetic evolution**.



Genetic Algorithm

Individual → Chromosome → Genes



Population (0) + Time (0) = Generation (0)

⋮

Population (i) + Time (i) = Generation (i)

?

Genetic Algorithm

- The basic idea behind the GA is the Darwinian principle of ***survival of the fittest*** among organisms threatened by predators and environmental hazards.
- The fittest members have a better chance of survival than others.
- They are more likely to adapt to evolving conditions, and ***their offspring may inherit*** their traits and learn their skills, thus producing even fitter future generations.
- Furthermore, genetic ***mutations*** occur randomly in members of species, and some of those mutations ***may improve*** the chances of long-term persistence of fit individuals and their evolutionary descendants.

Genetic Algorithm: Analogy

Optimization Algorithm	Genetic Algorithm
Decision Variable	Gene
Solution	Chromosome ≡ Individual
(N) Solutions/Iteration	Population Size
Max. Number of Iterations	Number of Generations
Objective Function	Fitness Function
Old Solution	Parent
New Solution	Off-spring
Best Solution	Elite
Selection of Solutions	Surviving Parents
Generation of Solutions	Genetic Operators

Genetic Algorithm: Solution Representation

- Chromosome Representation: ($\boldsymbol{n\ Genes}$)

$Gene_n$	...	$Gene_5$	$Gene_4$	$Gene_3$	$Gene_2$	$Gene_1$	$Gene_0$

Permutation Problem

Combination Problem

Binary Problem

Arithmetic Problem

- Generation Representation: ($\boldsymbol{m\ Chromosomes}$)

$\boldsymbol{x}_{\boxed{i}\ \boxed{j}}$

Generation (i)

Chromosome (j)

x_{i1}	g_n	...	g_5	g_4	g_3	g_2	g_1	g_0
x_{i2}	g_n	...	g_5	g_4	g_3	g_2	g_1	g_0
x_{i3}	g_n	...	g_5	g_4	g_3	g_2	g_1	g_0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
x_{im}	g_n	...	g_5	g_4	g_3	g_2	g_1	g_0

Genetic Algorithm: Fitness Function

- ***For a multi-objective function***

$$F(x) = \min\left(f_1 + f_2 + \frac{1}{f3}\right)$$

Minimize f_1 and f_2 & maximize f_3

- Evaluate the fitness value for each chromosome

Generated Solutions ***Fitness Values***

x_{i1}	g_n	$\cdots$	g_5	g_4	g_3	g_2	g_1	g_0	F_{i1}
x_{i2}	g_n	$\cdots$	g_5	g_4	g_3	g_2	g_1	g_0	F_{i2}
x_{i3}	g_n	$\cdots$	g_5	g_4	g_3	g_2	g_1	g_0	F_{i3}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_{im}	g_n	$\cdots$	g_5	g_4	g_3	g_2	g_1	g_0	F_{im}

Genetic Algorithm: Process

1. Select parents from old generation to mate.

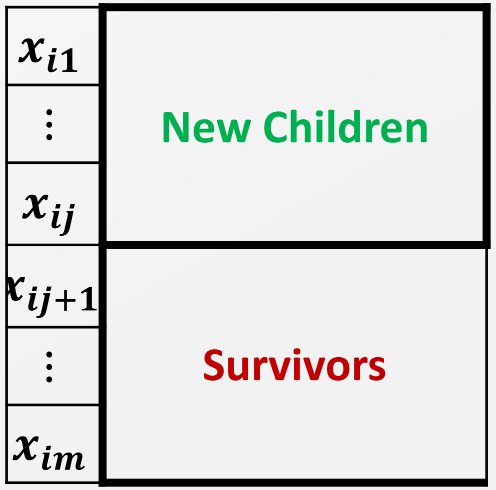
   (Parents Selection)

2. Perform GA operators from the selected parents to produce new off-springs.

   (Cross-Over & Mutation)

3. Which individuals are to be kicked out to be replaced by new children and which are to be kept in the next generation.

   (Survivor Selection)



- Tutorial 3 Recap
- **Genetic Algorithm**
- Application to Function Optimization
- Assembly Line Balancing Problem
- Techniques Comparison

Genetic Algorithm: Parent Selection

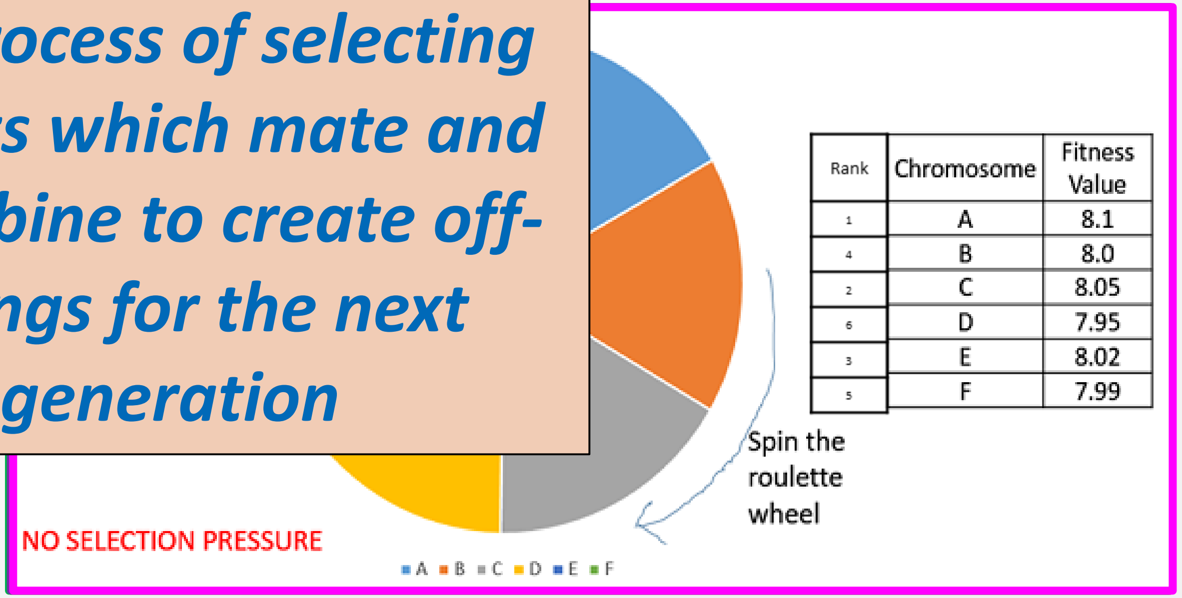
- **Fitness Proportionate Selection**
  - Roulette Wheel Selection
  - Stoc Univ Sampli
- **Tournament Selection**
- **Rank Selection**
- **Random Selection**

Randomly select parents existing population. There is n pressure towards fitter indiv therefore this strategy is usual

of the parents depends on the rank.

value.

The process of selecting parents which mate and recombine to create off-springs for the next generation



Rank	Chromosome	Fitness Value
1	A	8.1
4	B	8.0
2	C	8.05
6	D	7.95
3	E	8.02
5	F	7.99

Multiple Fixed Points → multiple parents

Genetic Algorithm: GA Operators

Elitism

- The chromosomes with the best fitness values.
- Elitism is usually applied in a GA with a **low** probability – p_e .

(Survivors)?

Cross-Over

- The crossover operator is analogous to **reproduction and biological crossover**.
- In each cross-over process, **2 parents** are selected to produce **2 off-springs** using the genetic material of the parents.
- Crossover is usually applied in a GA with a **high** probability – p_c .

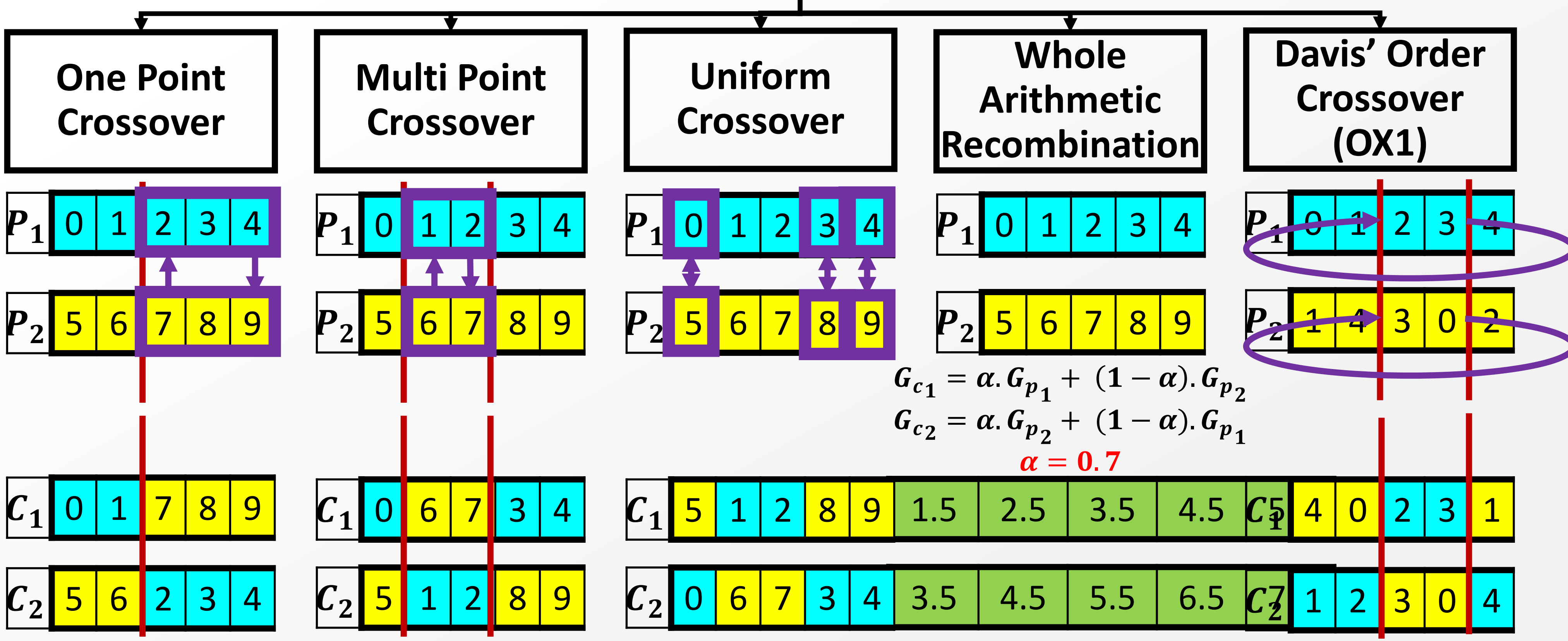
Mutation

- mutation may be defined as a **small random tweak** in the chromosome, to get a new solution.
- It is used to **maintain and introduce diversity** in the genetic population
- In each Mutation process, **1 parent** is selected to produce **1 child**.
- Mutation is usually applied with a **low** probability – p_m

$$\boxed{p_e + p_c + p_m \leq 100\%}$$

- Tutorial 3 Recap
- **Genetic Algorithm**
- Application to Function Optimization
- Assembly Line Balancing Problem
- Techniques Comparison

Genetic Algorithm: Cross-Over



Used for permutation based crossovers with the intention of transmitting information about relative ordering to the off-springs.

- Tutorial 3 Recap
- **Genetic Algorithm**
- Application to Function Optimization
- Assembly Line Balancing Problem
- Techniques Comparison

Genetic Algorithm: Mutation → *Exploration*

Bit Flip Mutation

P_1	1	0	1	1	0
C_1	1	0	1	0	0

Random Resetting

P_1	0	1	2	3	4
C_1	0	1	2	6	4

Swap Mutation

P_1	0	1	2	3	4
C_1	0	3	2	1	4

Scramble Mutation

P_1	0	1	2	3	4
C_1	0	2	3	1	4

Inversion Mutation

P_1	0	1	2	3	4
C_1	0	3	2	1	4

Select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.

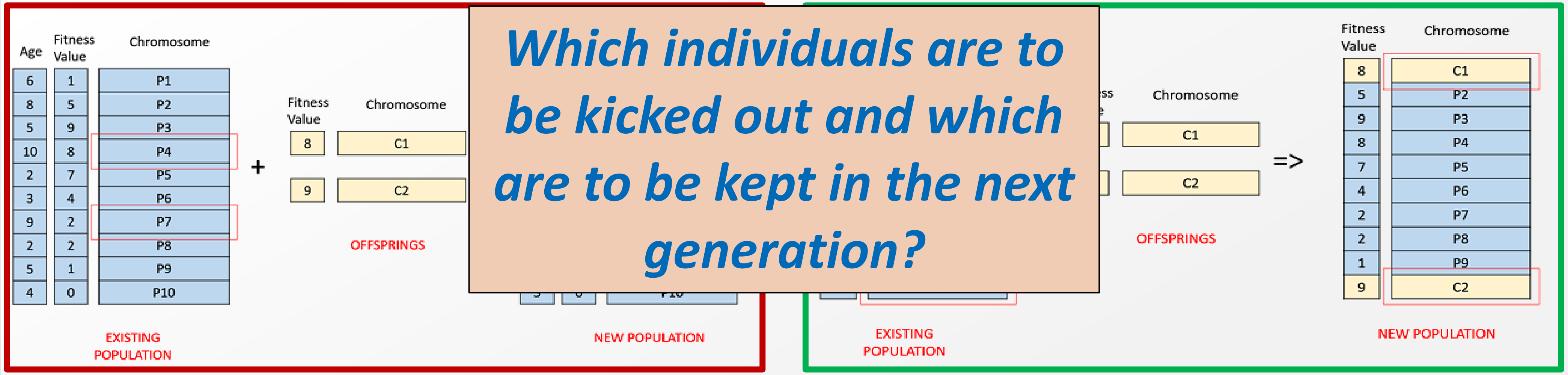
- Tutorial 3 Recap
- **Genetic Algorithm**
- Application to Function Optimization
- Assembly Line Balancing Problem
- Techniques Comparison

Genetic Algorithm: Survivor Selection

Age Based Selection

Fitness Based Selection

Which individuals are to be kicked out and which are to be kept in the next generation?



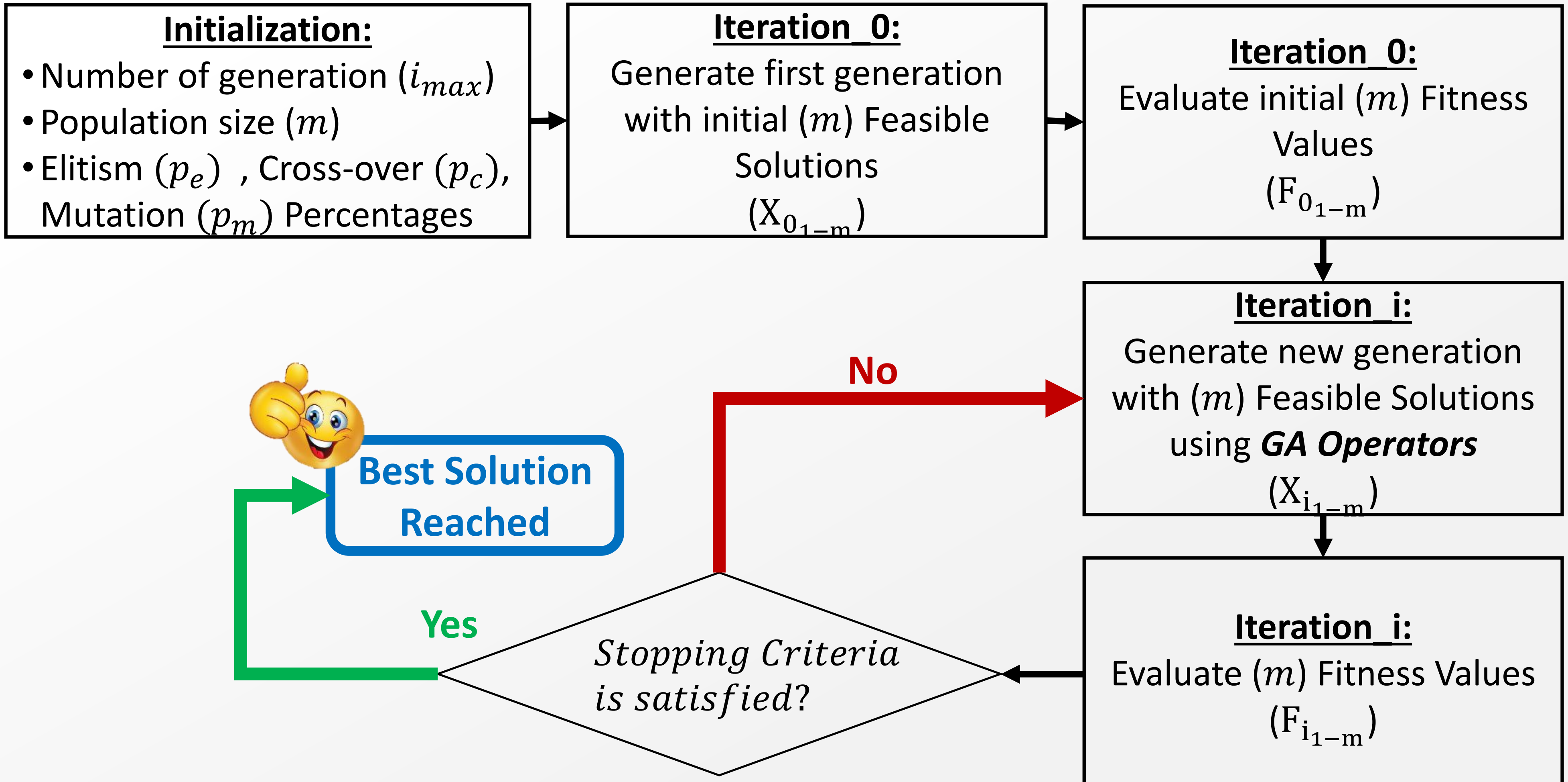
The newly generated children tend to kick-out the members with the worst fitness value. Thus, **the survivors from old generation are the *Elite* Members.**

recent Members.

Genetic Algorithm: Stopping Criteria

- When there has been no improvement in the population for X iterations.
- When we reach an absolute number of generations.
- When the objective function value has reached a certain pre-defined value.

Genetic Algorithm: Flow Chart



Genetic Algorithm: Algorithm

- Decision Variables: $\underline{\boldsymbol{x}} = x_1, x_2, \dots, x_p$
- Objective Function: $f(\underline{\boldsymbol{x}})$
- Initialize $\underline{\boldsymbol{x}}, i_{max}, m, p_e, p_c, p_m, x^*, f^*$

While $\underline{\boldsymbol{Stopping\ Criteria}}$ Ex. $\underline{(\boldsymbol{i \leq i_{max}})}$:

Generate random new population with (m) chromosomes x_{new}:

- p_e Elite members are survived from x_{prev}
- p_c Cross-over best members from x_{prev}
- p_m mutate worst members from x_{prev}

Compute fitness value: $f(x_{new})$

Update iteration counter: $i = i + 1$

Output best reached solution x^*, f^*

Case#1: Function Minimization using GA

- ***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 GA with the following parameters:***
  - Number of generations = 200
  - Population size = 5 **→ 5 chromosomes**
  - 20% elitism (as survivor selection) **→ 1 member**
  - 60% cross-over *(Elites parents selection & 1 point cross-over)* **→ 3 members**
  - 20% mutation *(Worst Member parent selection & Flip)* **→ 1 member**

- Perform 1 iteration only.

Case#1: Function Minimization using GA

- ***Solution Representation:***
  - Chromosome Representation: (Binary)

$X = 0$

$Gene_5$	$Gene_4$	$Gene_3$	$Gene_2$	$Gene_1$	$Gene_0$
0	0	0	0	0	0

$X = 32$

1	0	0	0	0	0

  - Generation Representation: (5 Chromosomes)

x_{i1}	g_5	g_4	g_3	g_2	g_1	g_0
x_{i2}	g_5	g_4	g_3	g_2	g_1	g_0
x_{i3}	g_5	g_4	g_3	g_2	g_1	g_0
x_{i4}	g_5	g_4	g_3	g_2	g_1	g_0
x_{i5}	g_5	g_4	g_3	g_2	g_1	g_0

Case#1: Function Minimization using GA

- ***Iteration_0:*** Initial Solution

x_{01}	1	0	0	0	0	0	$X = 32$	F_{01}	228	
x_{02}	0	1	0	0	0	1	$X = 17$	F_{02}	-2973	
x_{03}	0	0	0	1	0	0	$X = 4$	F_{03}	-2804	
x_{04}	0	0	1	0	1	0	$X = 10$	F_{04}	**-4100**	→ **Elite**
x_{05}	0	1	0	1	0	1	$X = 21$	F_{05}	-1801	

Case#1: Function Minimization using GA

- ***Iteration_1:***

x_{11}	0	0	1	0	1	0	→ **Elite of Iteration_0**
x_{12}							→ **Cross-over Members**
x_{13}							
x_{14}							
x_{15}	0	0	0	1	0	1	→ **Mutation Member**

Mutation Parent	1	0	0	0	0	0	→ **Worst of Iteration_0**
Mutation Child	0	0	0	1	0	1	

- Tutorial 3 Recap
- Genetic Algorithm
- **Application to Function Optimization**
- Assembly Line Balancing Problem
- Techniques Comparison

Case#1: Function Minimization using GA

- ***Iteration_1:*** Cont'd

x_{11}	0	0	1	0	1	0	→ **Elite of Iteration_0**
x_{12}	0	0	1	0	0	1	→ **Cross-over Members**
x_{13}	0	1	0	0	1	0	
x_{14}							
x_{15}	0	0	0	1	0	1	→ **Mutation Member**

Cross-over Parent1	0	0	1	0	1	0	→ **Elite of Iteration_0**
Cross-over Parent2	0	1	0	0	0	1	→ **2^{nd} Elite of Iteration_0**
Cross-over Child1	0	0	1	0	0	1	
Cross-over Child2	0	1	0	0	1	0	

- Tutorial 3 Recap
- Genetic Algorithm
- **Application to Function Optimization**
- Assembly Line Balancing Problem
- Techniques Comparison

Case#1: Function Minimization using GA

- ***Iteration_1:*** Cont'd

x_{11}	0	0	1	0	1	0	→ **Elite of Iteration_0**
x_{12}	0	0	1	0	0	1	→ **Cross-over Members**
x_{13}	0	1	0	0	1	0	
x_{14}	0	0	1	1	0	0	
x_{15}	0	0	0	1	0	1	→ **Mutation Member**

Cross-over Parent1	0	0	1	0	1	0	→ **Elite of Iteration_0**
Cross-over Parent2	0	0	0	1	0	0	→ **3rd Elite of Iteration_0**
Cross-over Child1	**0**	**0**	**1**	**1**	**0**	**0**	**Randomly Select one of the off-springs to survive**
Cross-over Child2	0	0	0	0	1	0	

Case#1: Function Minimization using GA

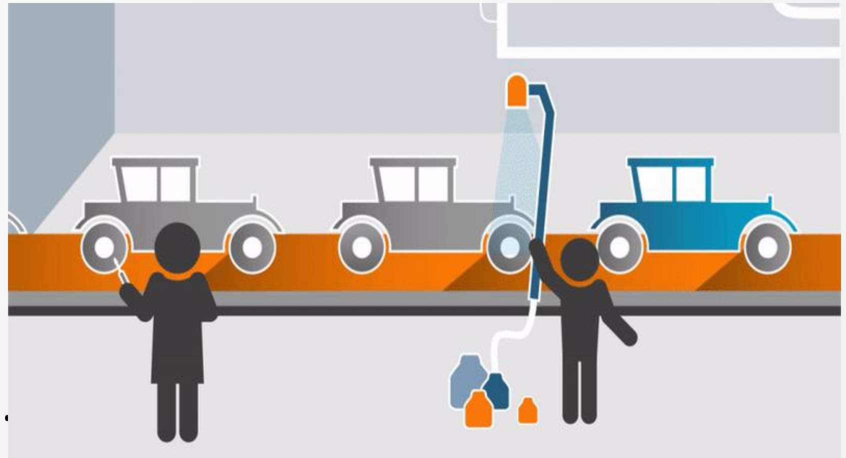
- ***Iteration_1:*** Cont'd

x_{11}	0	0	1	0	1	0	$X = 10$	F_{11}	**-4100**	→ **Elite**
x_{12}	0	0	1	0	0	1	$X = 9$	F_{12}	-4069	
x_{13}	0	1	0	0	1	0	$X = 18$	F_{13}	-2692	
x_{14}	0	0	1	1	0	0	$X = 12$	F_{14}	-3988	
x_{15}	0	0	0	1	0	1	$X = 5$	F_{15}	-3225	

Continue iterating with the same manner ….

Case#2: 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.

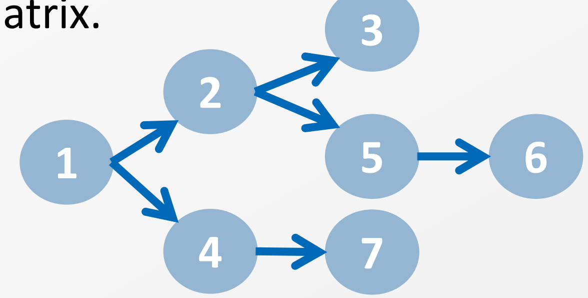


- **Constraint Equation and Inequalities:**
  - Sequential order of tasks (i.e. precedence constraints).
  - Cycle time of the assembly line.
  - Maximum number of workstations.

- Tutorial 3 Recap
- Genetic Algorithm
- Application to Function Optimization
- **Assembly Line Balancing Problem**
- Techniques Comparison

Case#2: Assembly Line Balancing

- ***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.

$Task\ \#i$

$General\ Solution \rightarrow$	T_i	T_i	T_i	...	...	T_i	-	-
	WS_1	WS_2		WS_j		WS_{Nu}		WS_m

$\#\ Used\ WS$

- **Example:**

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

Assembly Line Balancing: Formulation

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

$$\min f = \min N_u$$

OR

  - 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

- ***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.
    $$N_u \leq m = 6$$

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

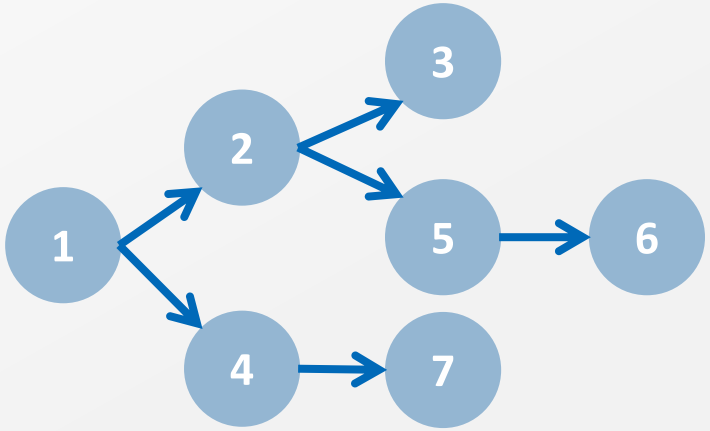
$N_u = 6$

$Solution\ 1 \rightarrow$

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

Assembly Line Balancing: GA

- ***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 GA with the following parameters:***
  - Number of generations = 200
  - Population size = 4 **→ 4 chromosomes**
  - 20% elitism (as survivor selection) **→ 1 member**
  - 60% cross-over *(Elites parents selection & Davis' Order Crossover)* **→ 2 members**
  - 20% mutation *(Worst Member parent selection & swapping)* **→ 1 member**

Assembly Line Balancing: GA

- ***Solution Representation:***
  - Chromosome Representation: (Permutation)

$Gene_5$	$Gene_4$	$Gene_3$	$Gene_2$	$Gene_1$	$Gene_0$
Ti	Ti	Ti	Ti	Ti	Ti

  - Generation Representation: (4 Chromosomes)

x_{i1}	g_5	g_4	g_3	g_2	g_1	g_0
x_{i2}	g_5	g_4	g_3	g_2	g_1	g_0
x_{i3}	g_5	g_4	g_3	g_2	g_1	g_0
x_{i4}	g_5	g_4	g_3	g_2	g_1	g_0

Assembly Line Balancing: GA

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

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

- ***Iteration_0:*** Initial Solution

x_{01}	1	2	3	4,5	6	7	✓
x_{02}	1,2	3,4	5	6	7		✓
x_{03}	1,4	2	3	5	6	7	✓
x_{04}	1	2,4	3	5	6	7	✓

Feasibility Check

Tws_{01}	1	5	4	**8**	6	5
Tws_{02}	6	**7**	5	6	5	
Tws_{03}	4	5	4	5	**6**	5
Tws_{04}	1	**8**	4	5	6	5

- Tutorial 3 Recap
- Genetic Algorithm
- Application to Function Optimization
- **Assembly Line Balancing Problem**
- Techniques Comparison

Assembly Line Balancing: GA

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

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

- ***Iteration_0:*** Initial Solution

x_{01}	1	2	3	4,5	6	7	F_{01}	3.807
x_{02}	1,2	3,4	5	6	7		F_{02}	1.414
x_{03}	1,4	2	3	5	6	7	F_{03}	**1.354** → **Elite**
x_{04}	1	2,4	3	5	6	7	F_{04}	3.807

Feasibility Check

Tws_{01}	1	5	4	**8**	6	5
Tws_{02}	6	**7**	5	6	5	
Tws_{03}	4	5	4	5	**6**	5
Tws_{04}	1	**8**	4	5	6	5

Assembly Line Balancing: GA

- ***Iteration_1:***

x_{11}	1,4	2	3	5	6	7
x_{12}						
x_{13}						
x_{14}	1	2	5	3,4	6	7

→ Elite of Iteration_0

→ Cross-over Members

→ Mutation Member

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

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

Mutation Parent	1	2	$\boxed{3}$	4,$\boxed{5}$	6	7
Mutation Child	1	2	5	3,4	6	7

→ Worst of Iteration_0

- Tutorial 3 Recap
- Genetic Algorithm
- Application to Function Optimization
- **Assembly Line Balancing Problem**
- Techniques Comparison

Assembly Line Balancing: GA

- ***Iteration_1:*** Cont'd

x_{11}	1,4	2	3	5	6	7
x_{12}	1,2	3	5	6	7	4
x_{13}	1,4	3	5	6	7	2
x_{14}	1	2	5	3,4	6	7

→ Elite of Iteration_0

→ Cross-over Members

→ Mutation Member

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

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

Cross-over Parent1	1,2	3,4	5	6	7	
Cross-over Parent2	1,4	2	3	5	6	7
Cross-over Child1	1,2	3	5	6	7	4
Cross-over Child2	1,4	3	5	6	7	2

→ 2nd Elite of Iteration_0

→ Elite of Iteration_0

- Tutorial 3 Recap
- Genetic Algorithm
- Application to Function Optimization
- **Assembly Line Balancing Problem**
- Techniques Comparison

Assembly Line Balancing: GA

- ***Iteration_1:*** Cont'd

x_{11}	1,4	2	3	5	6	7	✓
x_{12}	1,2	3	5	6	7	4	✗
x_{13}	1,4	3	5	6	7	2	✗
x_{14}	1	2	5	3,4	6	7	✓

Feasibility Check

Tws_{11}	4	5	4	5	**6**	5
Tws_{12}	5	4	5	**6**	5	3
Tws_{13}	4	4	5	**6**	5	5
Tws_{14}	1	5	5	**7**	6	5

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

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

Assembly Line Balancing: GA

- ***Iteration_1:*** Cont'd

x_{11}	1,4	2	3	5	6	7	→ Elite of Iteration_0
x_{12}	1,2	3,4	5	6	7		→ Cross-over Members
x_{13}	1,4	2	5	6	7	3	→ Cross-over Members
x_{14}	1	2	5	3,4	6	7	→ Mutation Member

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

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

Cross-over Parent1	1,2	3,4	5	6	7	
Cross-over Parent2	1,4	2	3	5	6	7
Cross-over Child1	1,2	3,4	5	6	7	
Cross-over Child2	1,4	2	5	6	7	3

→ 2nd Elite of Iteration_0

→ Elite of Iteration_0

- Tutorial 3 Recap
- Genetic Algorithm
- Application to Function Optimization
- **Assembly Line Balancing Problem**
- Techniques Comparison

Assembly Line Balancing: GA

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

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

- ***Iteration_1:*** Cont'd

x_{11}	1,4	2	3	5	6	7
x_{12}	1,2	3,4	5	6	7	
x_{13}	1,4	2	5	6	7	3
x_{14}	1	2	5	3,4	6	7

Feasibility Check

Tws_{11}	4	5	4	5	6	5
Tws_{12}	6	7	5	6	5	
Tws_{13}	4	5	5	6	5	4
Tws_{14}	1	5	5	7	6	5

- Tutorial 3 Recap
- Genetic Algorithm
- Application to Function Optimization
- **Assembly Line Balancing Problem**
- Techniques Comparison

Assembly Line Balancing: GA

- ***Iteration_1:*** Cont'd

x_{11}	1,4	2	3	5	6	7	F_{11}	**1.354**	→ Elite
x_{12}	1,2	3,4	5	6	7		F_{12}	1.414	
x_{13}	1,4	2	5	6	7	3	F_{13}	**1.354**	→ Elite
x_{14}	1	2	5	3,4	6	7	F_{14}	2.8577	

Feasibility Check

Tws_{11}	4	5	4	5	**6**	5
Tws_{12}	6	**7**	5	6	5	
Tws_{13}	4	5	5	**6**	5	4
Tws_{14}	1	5	5	**7**	6	5

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

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

Optimization Techniques Comparison

	Simulated Annealing (SA)	Genetic Algorithm (GA)
Technique Category	Trajectory-based	Population-based
Inspiration	Physical Annealing of Metals	Genetic Evolution of Living Creatures
Solution(s) Representation	1 Solution/iteration	• Each iteration is composed of **(1) Population** • Each population has **(N) Chromosomes** • Each Chromosome has **(M) Genes**
Generation of New Solution Criteria	Generation Criteria is based on the solution representation • **Ex.:** Swapping **or** Remove and insert in case of permutation sol rep	GA Operators • Elitism • Cross-Over • Mutation

Optimization Techniques Comparison

	Simulated Annealing (SA)	Genetic Algorithm (GA)
Convergence Criteria	Temperature Cooling Criteria • **Linear** $T_i = T_0 - \beta i$ • **Geometry** $T_i = T_0 \alpha^i$	Number of Generations
Acceptance Criteria	• If new sol is **better**, accept new solution • If new sol is **worse**, based on **probability** $P = e^{-\frac{\lvert -\Delta F \rvert}{T_i}}$,and **random num** r $\begin{cases} P < r & \textit{reject Sol} \\ P > r & \textit{accept Sol} \end{cases}$	All Solutions in the populations are **accepted** as long as: • Solutions generated are **feasible.** • **No duplicate** solutions in the population specially in the early iterations to avoid exploitation.
Stopping Criteria	• Reaching Final Temperature • Final Number of Iterations	• Final number of generations • Convergence to one solution • Elite is the same across many generations

See you Next time … ☺