

**Short Term Training Programme
On
Ethical AI/ML for Innovation: Exploring Technologies and
Applications**

24 February – 07 March, 2025

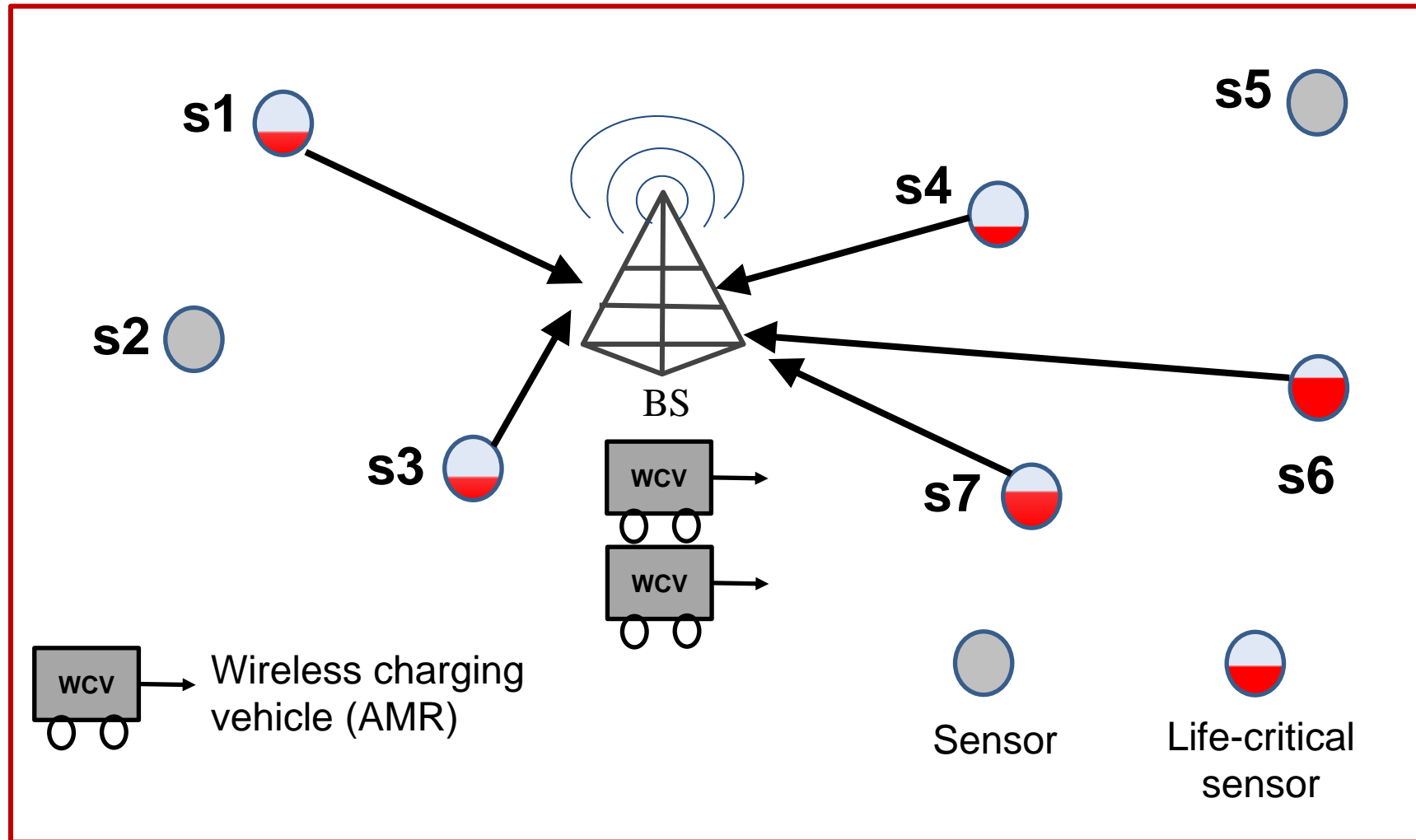


**Department of Artificial Intelligence
SVNIT, Surat**

Path Planning of Autonomous Mobile Robot (AMR)

by
Rohit Kumar

Path Planning of AMR: Wireless Rechargeable Sensor Networks



Path Planning of AMR: Wireless Rechargeable Sensor Networks

Objective: Minimize the total energy consumed for WCVs travel

Constraints:

- Tours of any two WCVs are non-overlapping
- Each sensor is charged at most once in a charging tour

Proposed Algorithm

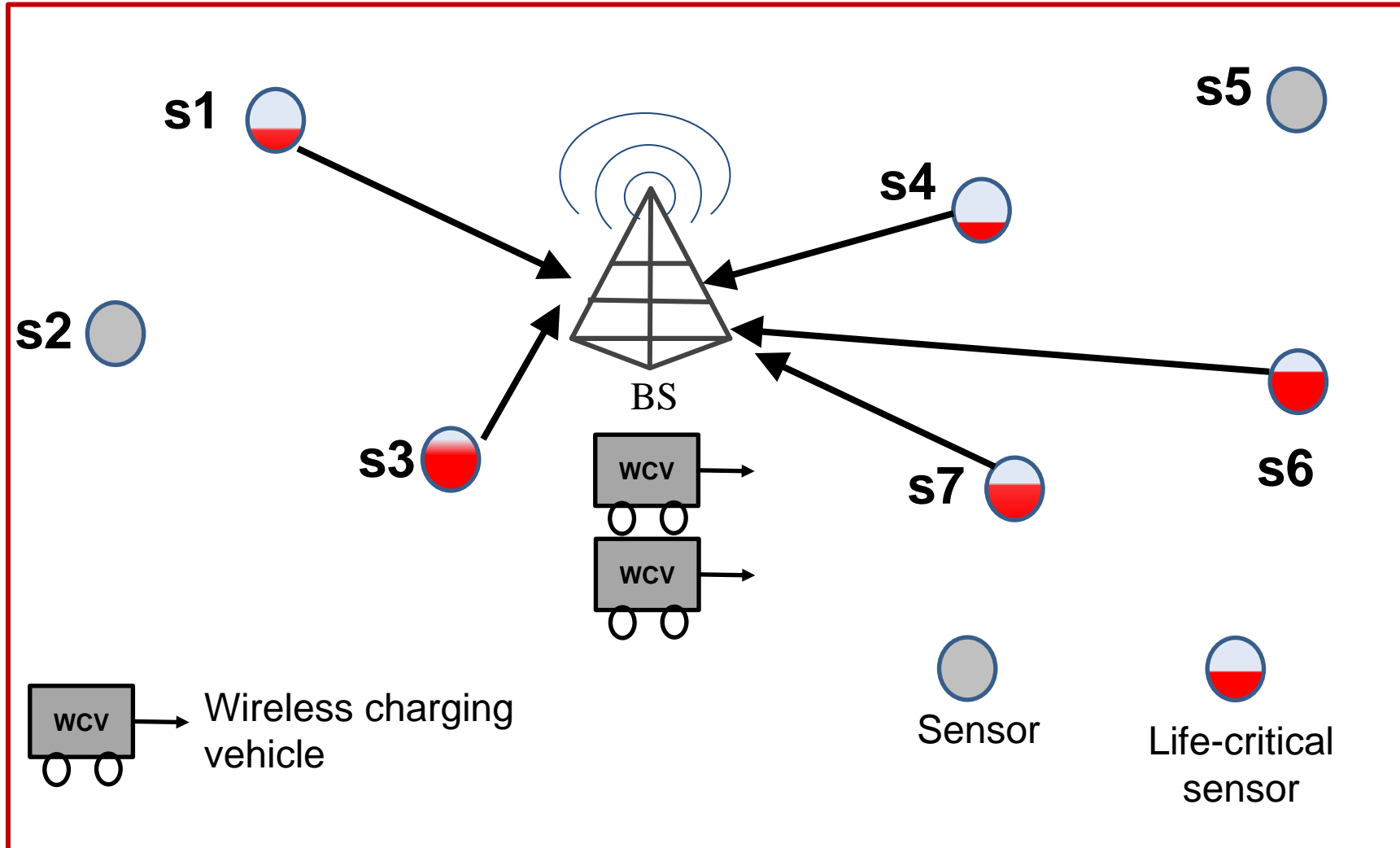
The algorithm is divided into two stages:

Stage I: Clustering the sensors using **k-means clustering** [k = No of WCVs]

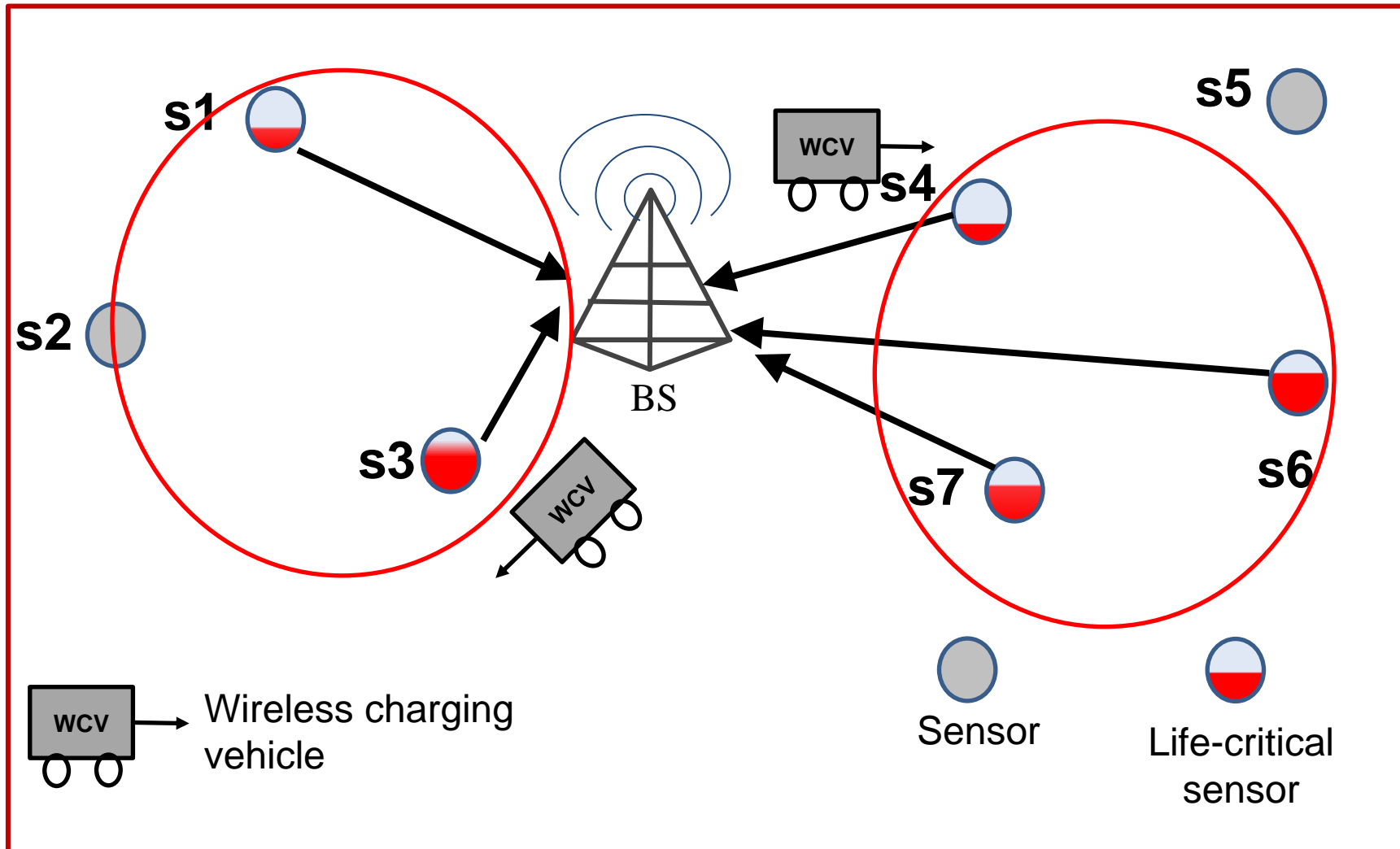
Stage II: Determining routes for WCVs for charging sensors in their regions.

Algorithm ??????

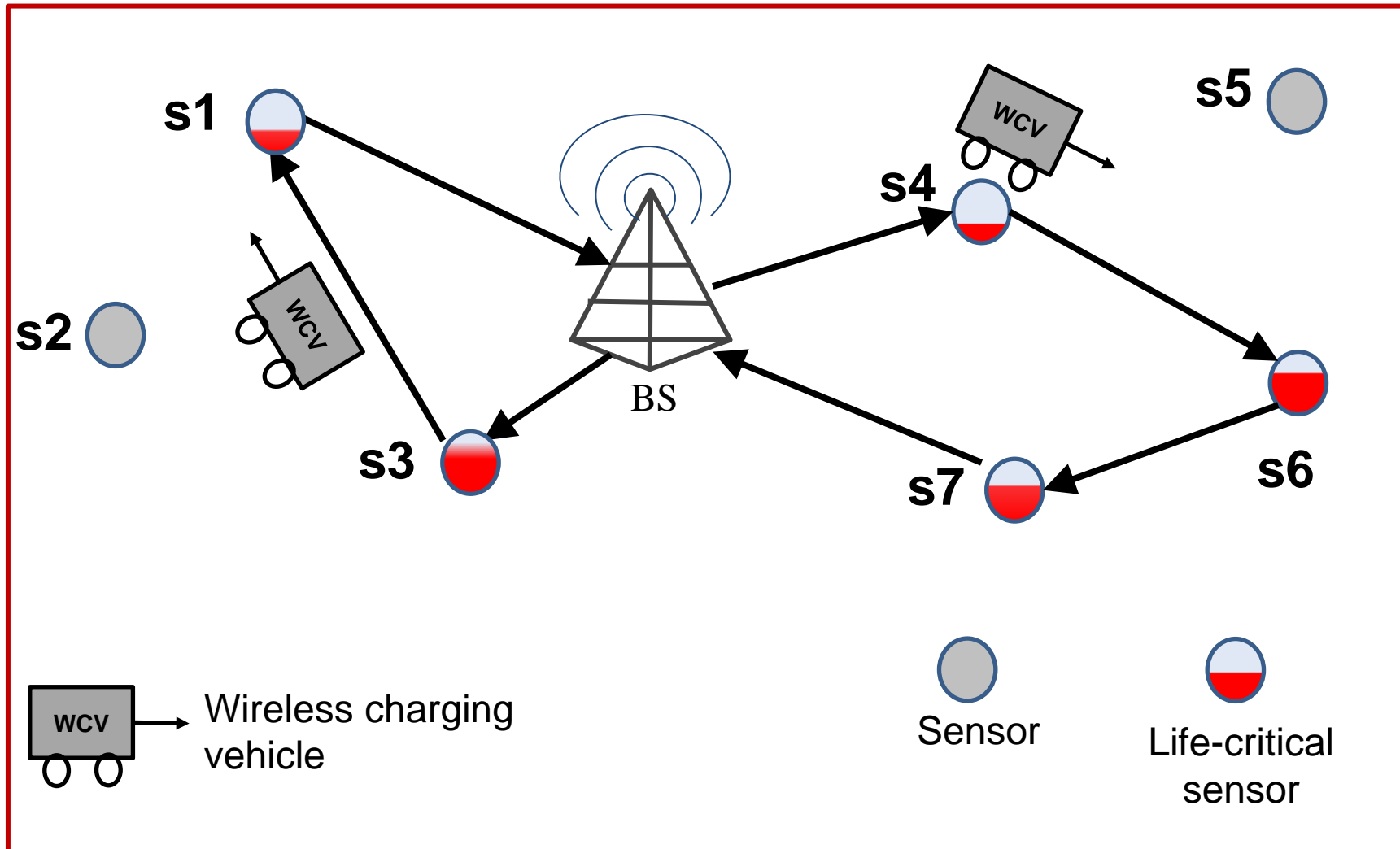
Proposed Algorithm



Stage 1: Distribution of Charging Request



Stage 2: Determining the Tour of WCVs



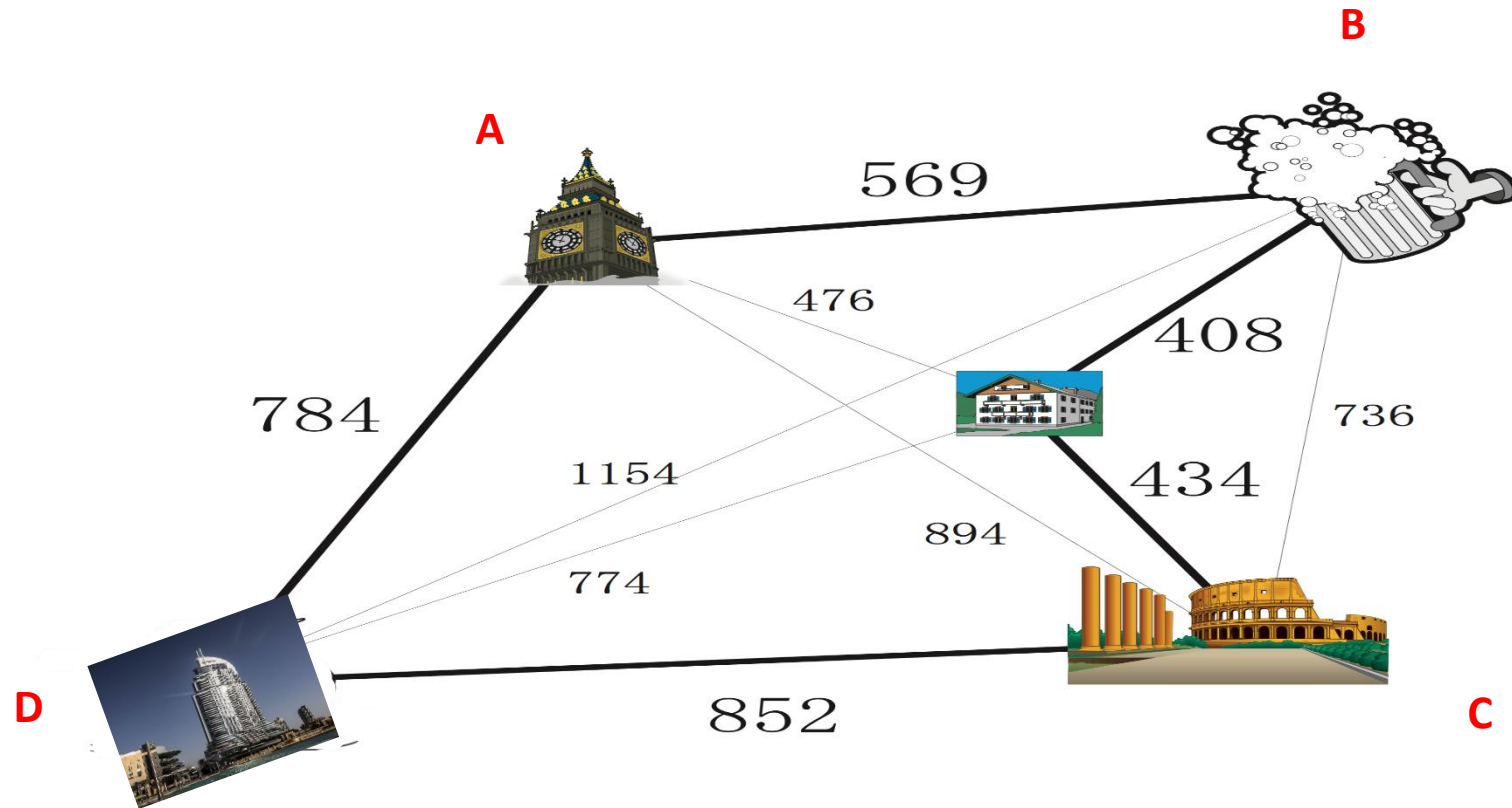
How can the tour for the WCV be determined efficiently?

Determining the Tour for the WCV?

- This problem is similar to the Traveling Salesman Problem (TSP).
- So, **what is TSP?**

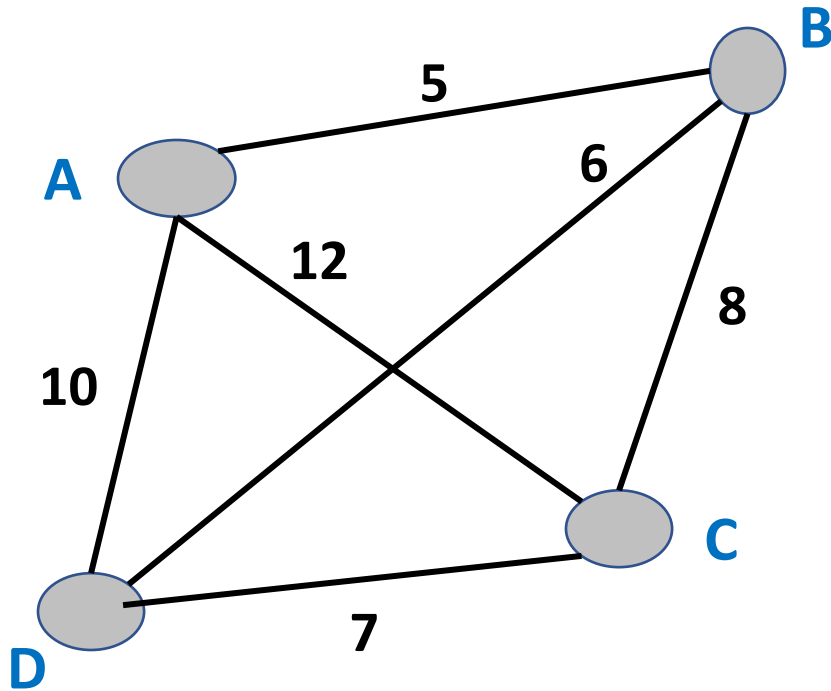
Travelling Salesman Problem (TSP):

Statement: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?"



TSP

Example with 4 cities



Possible TSP solutions:

Let's fix **City A** as the starting point

1. $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$ Total distance = $5+8+7+10 = 30$)
2. $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A$ Total distance = $5+6+7+12 = 30$)
3. $A \rightarrow C \rightarrow B \rightarrow D \rightarrow A$ Total distance = $12+8+6+10 = 36$)
4. $A \rightarrow C \rightarrow D \rightarrow B \rightarrow A$ Total distance = $12+7+6+5 = 30$)
5. $A \rightarrow D \rightarrow B \rightarrow C \rightarrow A$ Total distance = $10+6+8+12 = 36$)
6. $A \rightarrow D \rightarrow C \rightarrow B \rightarrow A$ Total distance = $10+7+8+5 = 30$)

TSP

Num of Cities (n)	Total Possible Tours	Total time taken (Assume time taken by 1 tour is 1 milisecond (0.001 second)
n = 4	$4! = 24$	24 milisecond = 0.024 seconds
n = 10	$10! = 3628800$	3628800 milliseconds = 3628.8 seconds = 60.48 minutes
n=20	$20! = 2432902008176640000$	2432902008176640000 milisecond = 2432902008176640 seconds = 7.72×10^{10} years
n = 40	$40! = 8.159152832478977 \times 10^{47}$	$8.159152832478977 \times 10^{44}$ milisecond = 2.59×10^{39} years
n = 50	$50! = 9.332621544394415 \times 10^{157}$	$9.332621544394415 \times 10^{157}$ seconds = 2.95×10^{150} years

TSP

Why TSP with n cities cannot be solved in polynomial time?

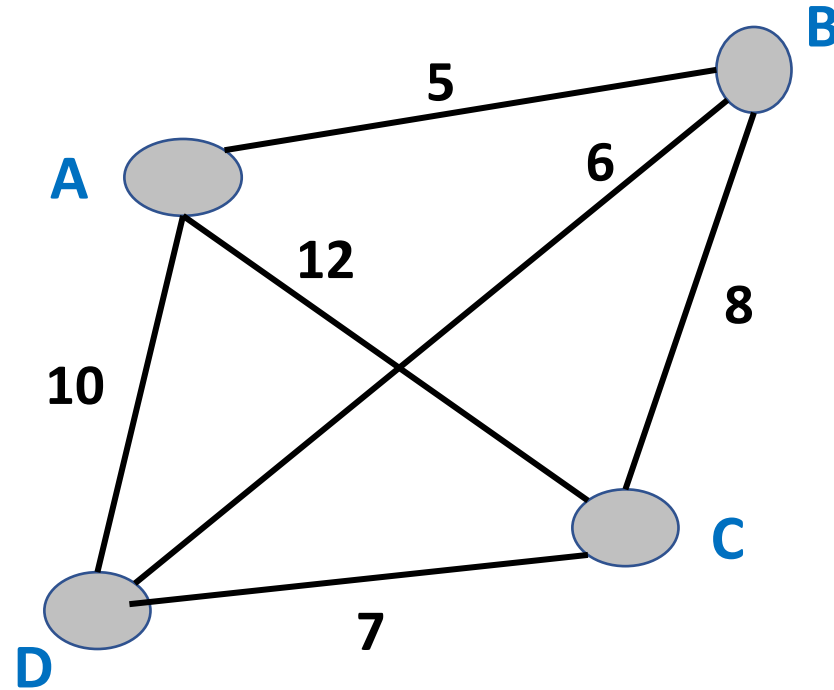
- If there are n cities, there are $n!$ possible tours of the cities.
- To find the length of a tour will take $O(n)$ time.
- Thus, the total time taken to find the shortest tour (optimal tour) is $O(n!) * O(n) = O(n!)$, **which is not polynomial**
- Since optimal tour **cannot be obtained in TSP**, Genetic Algorithm can be used to solve it.



Rohit Kumar



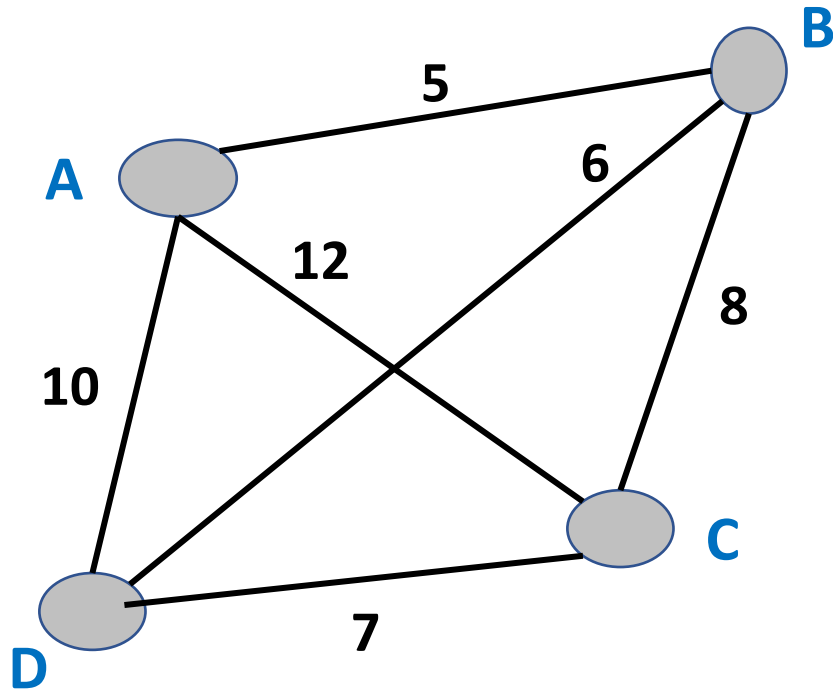
Solution to the TSP using Approximation Algorithms



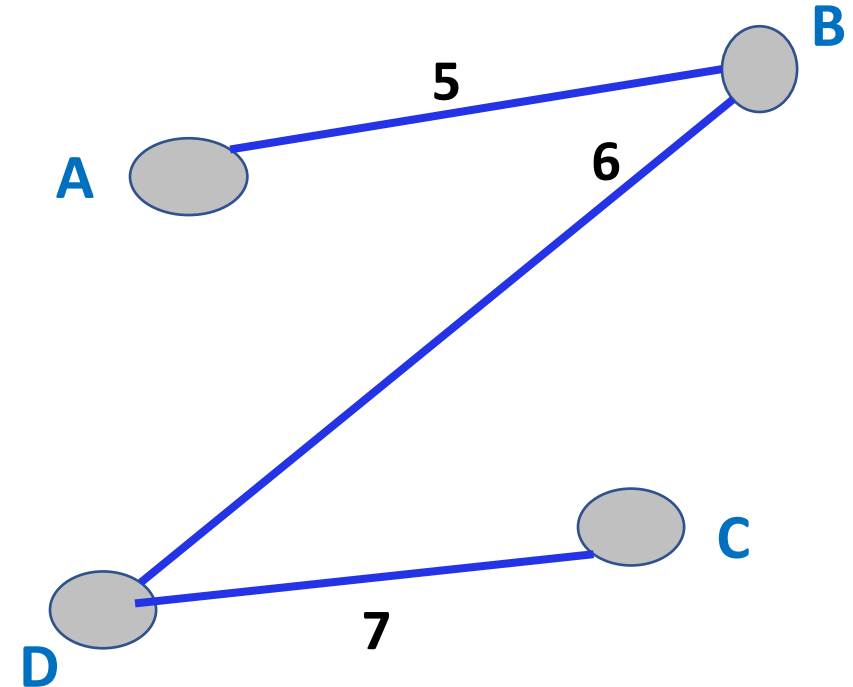
Graph $G = (V, E)$

Solution to the TSP using Traditional Algorithms

Step 1: Minimum Spanning Tree (MST) of Graph G

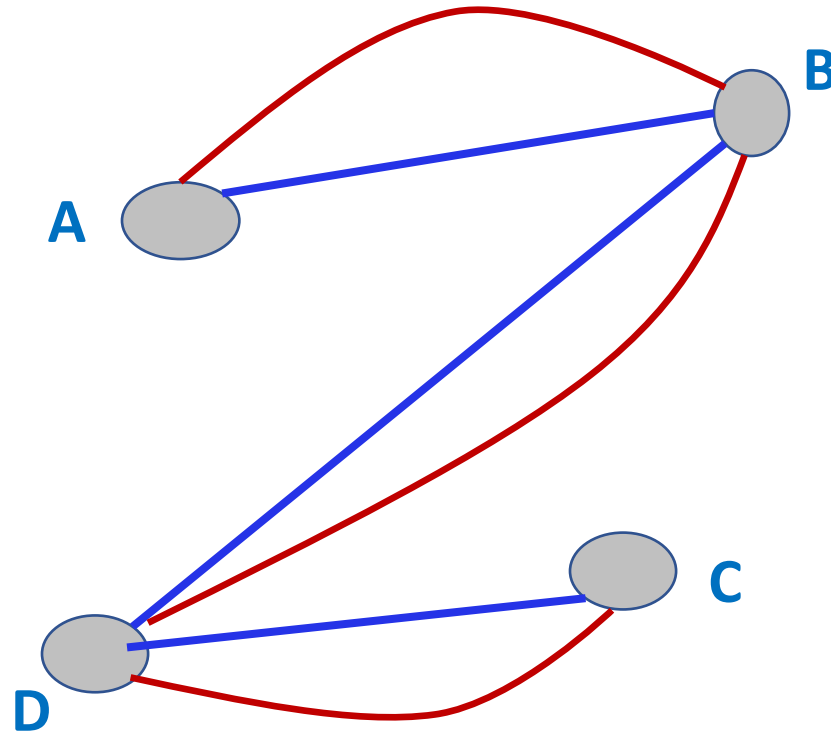


Graph $G = (V, E)$



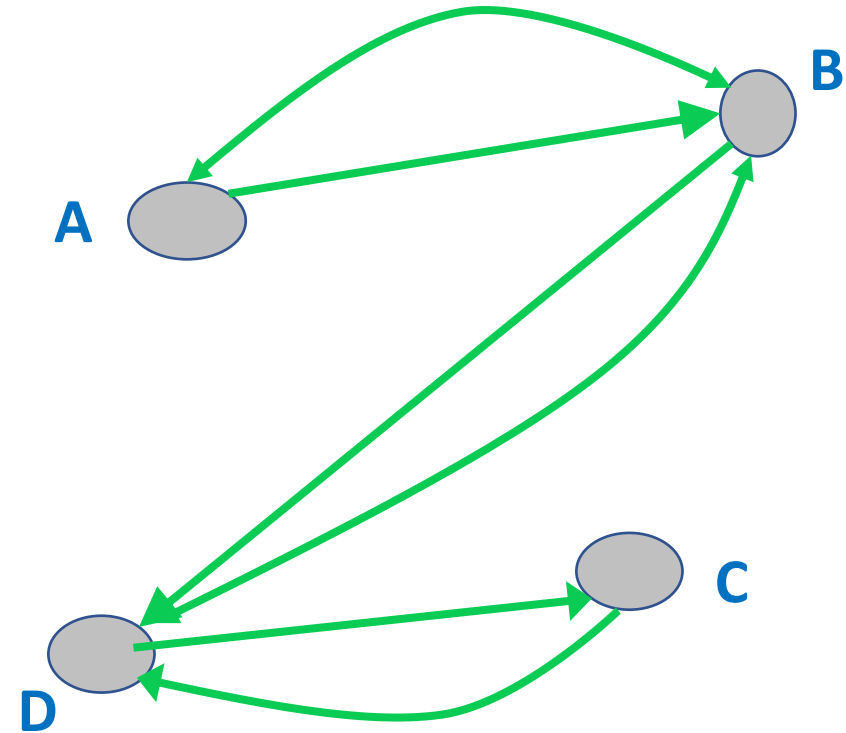
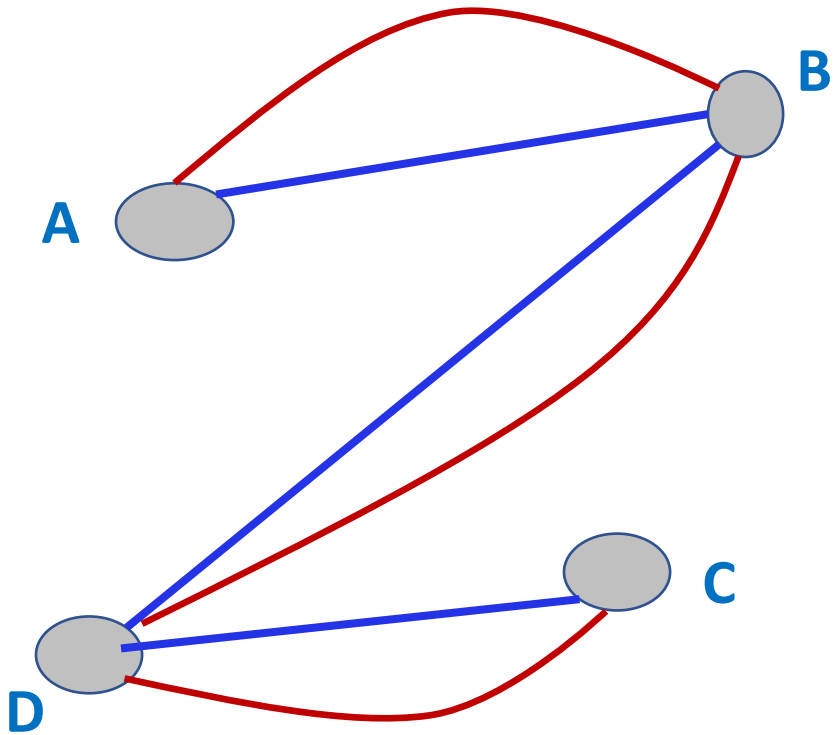
Solution to the TSP using Traditional Algorithms

Step 2: Duplicate the Edges of MST



Solution to the TSP using Traditional Algorithms

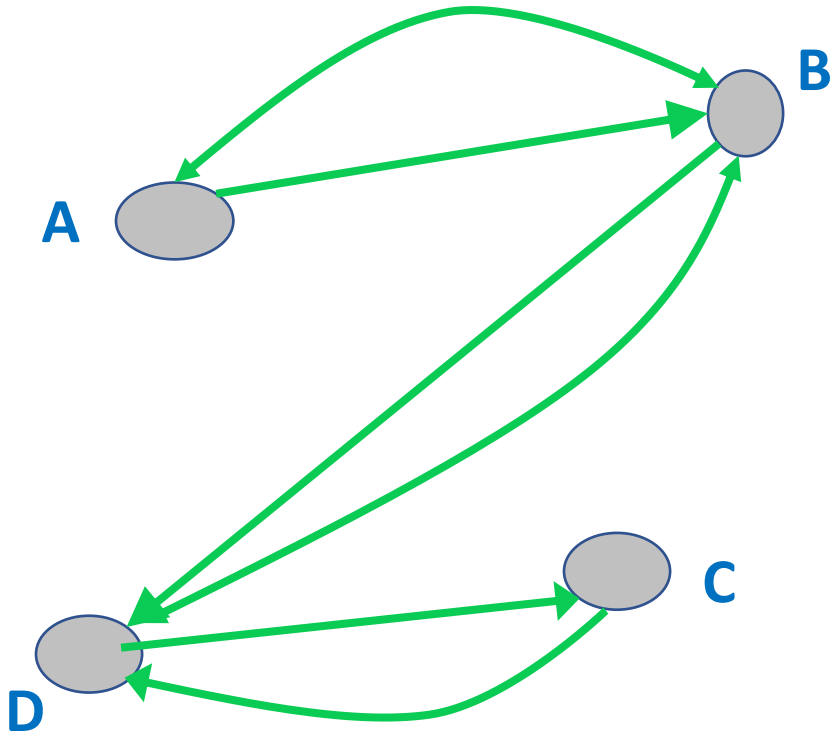
Step 3: Find an Euler Tour



Euler Tour $A \rightarrow B \rightarrow D \rightarrow C \rightarrow D \rightarrow B \rightarrow A$

Solution to the TSP using Traditional Algorithms

Step 3: Removing the vertex which are repeated



Euler Tour $A \rightarrow B \rightarrow D \rightarrow C \rightarrow D \rightarrow B \rightarrow A$

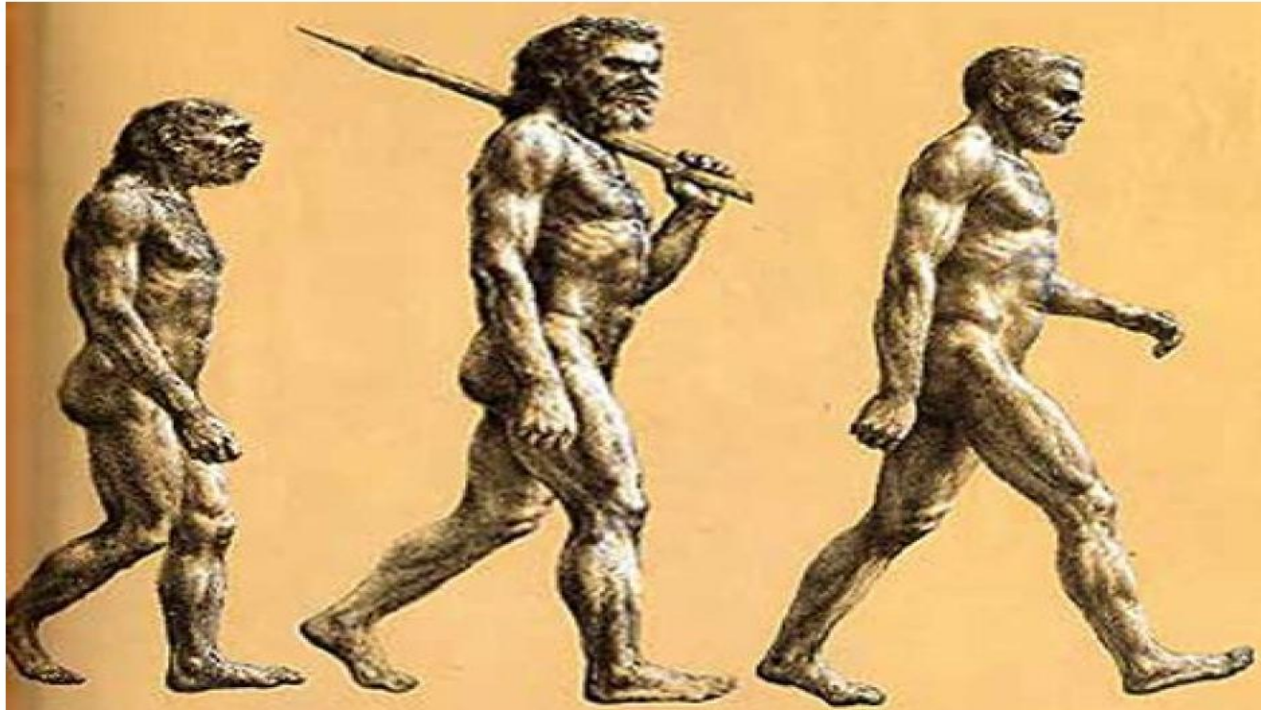
Final Solution TSP (Tour) $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A$

TSP Solution using Genetic Algorithms

Genetic Algorithms (GAs)

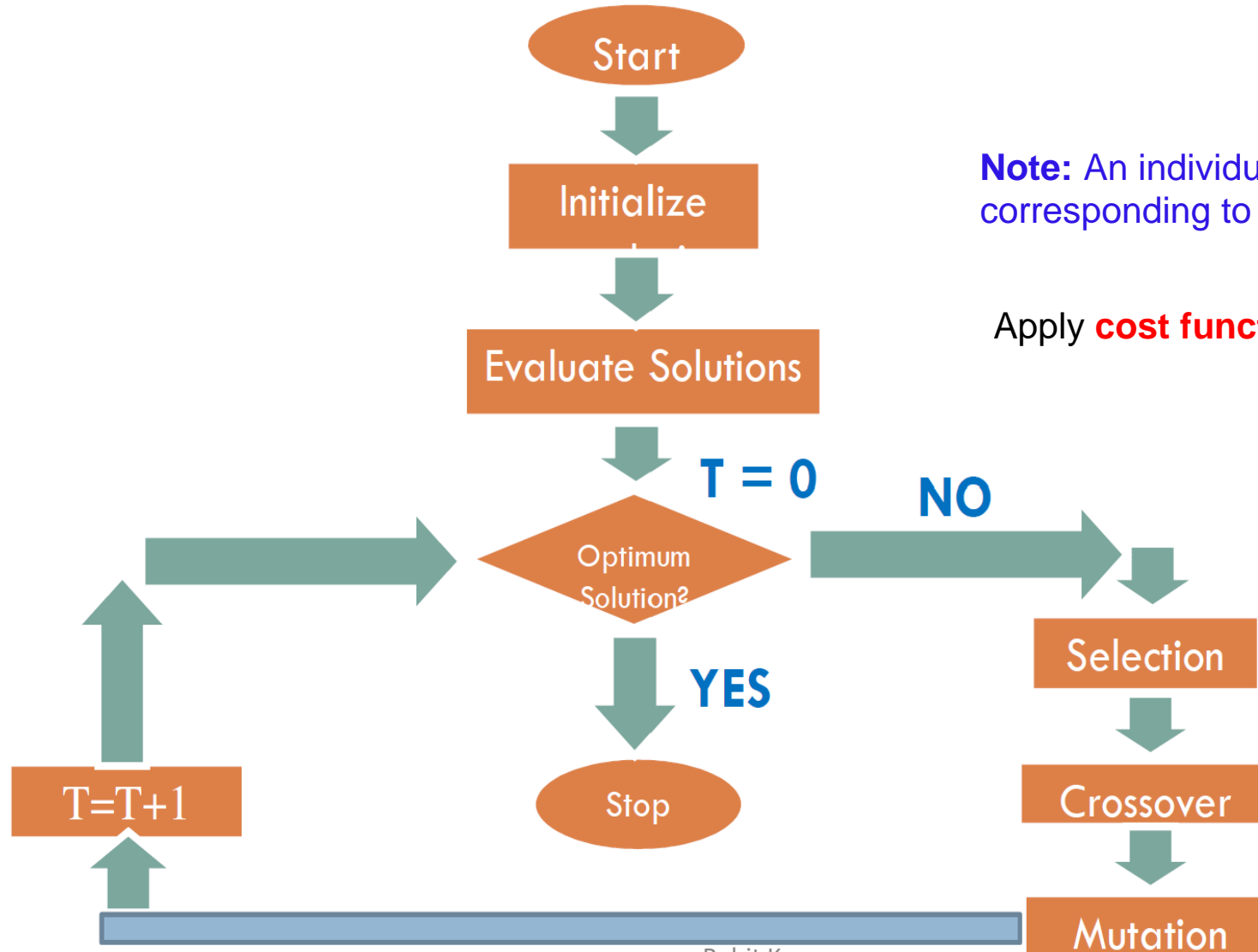
- Genetic Algorithms are heuristic search algorithms inspired by Darwin's Theory of Evolution in nature.
- GA's are a part of evolutionary computing, a branch of artificial intelligence
- GA's generate high-quality solutions for optimization problems in complex, often high-dimensional search spaces.
- GAs are inspired by the **process of natural selection**, reproduction and mutation
 - Which means those species that can adapt to changes in their environment can survive and reproduce and go to the next generation
 - They simulate “survival of the fittest” among individuals of consecutive generations to solve a problem

Genetic Algorithms (GAs)



**“Select The Best,
Discard The Rest”**

Simple Genetic Algorithm (GA)



Note: An individual in the population is corresponding to a possible solution

Apply **cost function** to each of the population

Simple Genetic Algorithm (GA)

```
function simpleGeneticAlgorithm (){  
    Initialize population;  
    Calculate fitness function;  
    While(fitness value != termination criteria){  
        Selection;  
        Crossover;  
        Mutation;  
        Calculate fitness value;  
    }  
}
```


GA Operators and Parameters

1. Selection
2. Crossover
3. Mutation

1. Selection

- The primary objective of the selection operator is to prioritize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.
- The **mating pool** consists of a subset of selected individuals (good solutions) from a population chosen after the selection operator.

“Selects the best, discards the rest”

Function of Selection Operator

- Identify the good solutions in a population
- Make multiple copies of the good solutions
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population

Function of Selection Operator

- Identify the good solutions in a population
- Make multiple copies of the good solutions
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population

how to identify the good solutions?

Fitness Function

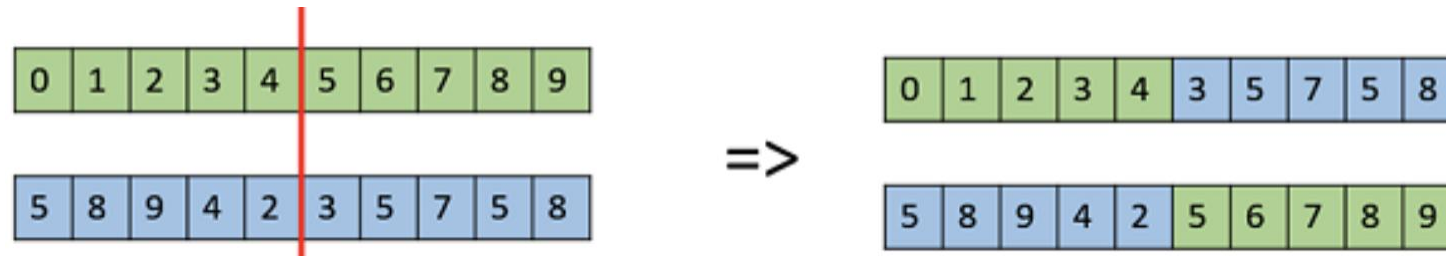
- A fitness value can be assigned to evaluate the solutions
- A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions
- A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem

2. Crossover

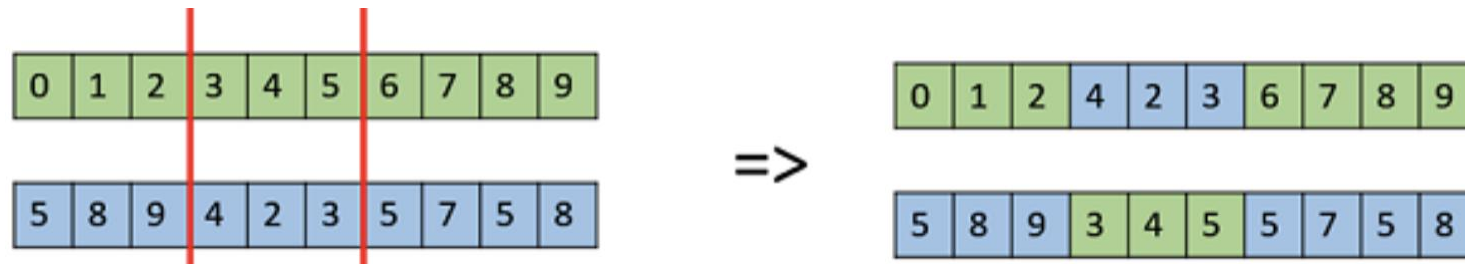
- The **crossover operator** is used to **create new solutions** from the existing solutions available in the mating pool after applying selection operator
- This operator exchanges the gene information between the solutions in the mating pool

Crossover Types

- **Single-Point Crossover:** A random crossover point is selected and the tails of its two parents are swapped to get new off-springs.

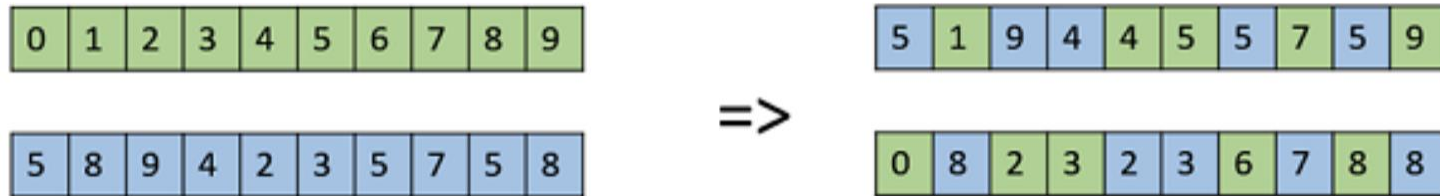


- **Two-Point Crossover:** Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



Crossover Types

- **Uniform Crossover:** In a uniform crossover, **we don't divide the chromosome into segments**, rather we treat each gene separately.
 - In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring (new solution).

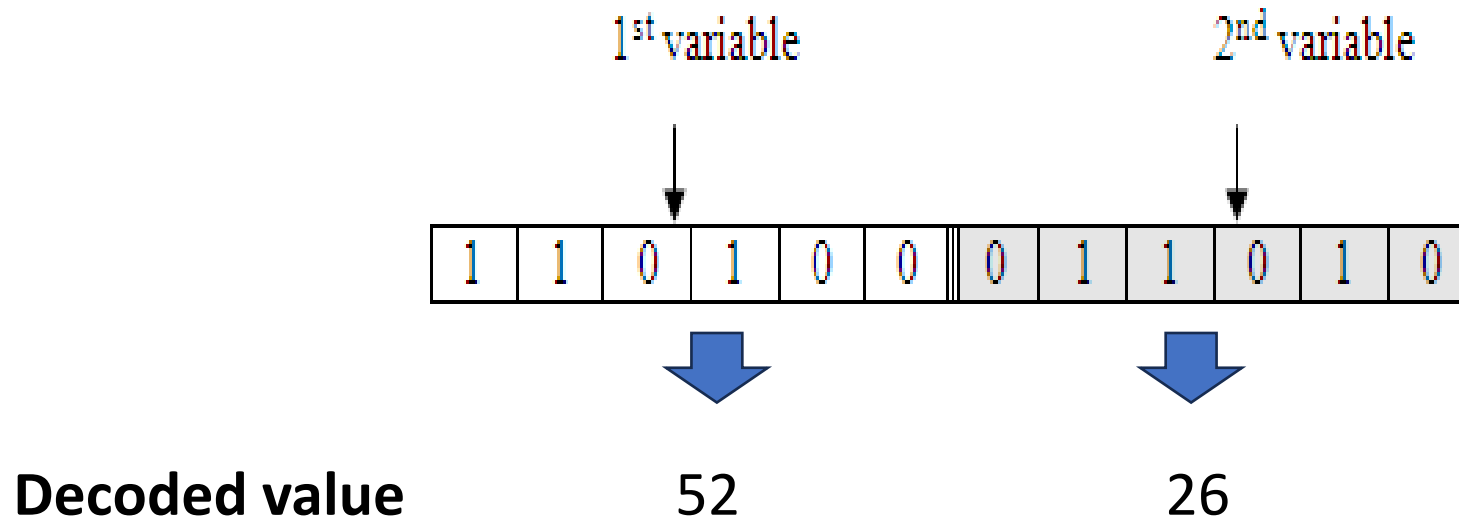


Encoding

- The process of representing a solution in the form of a string that conveys the necessary information.
- Each gene in a chromosome determines a specific characteristic of an individual, just as each bit in the string represents a particular trait (attribute) of the solution.

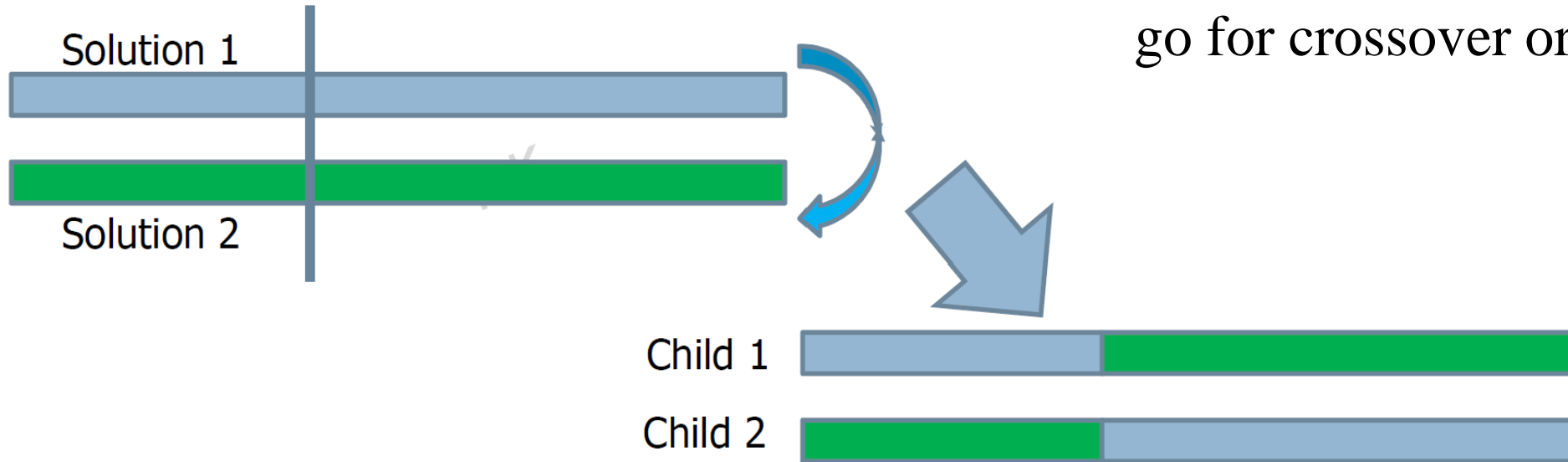
Encoding Methods

- Most common method of encoding is **binary coded**.
- Chromosomes are strings of 1 and 0 and each position in the chromosome represents a particular characteristic of the problem



2. Crossover Operator

- The most popular crossover selects any two solutions strings randomly from the mating pool and some portion of the strings is exchanged between the strings.
- The selection point is selected randomly.
- A probability of crossover is also introduced in order to give freedom to an individual solution string to determine whether the solution would go for crossover or not.



Crossover: Example

Single-Point Crossover

Let's assume we have two parent binary strings of length 8:

Parent 1: 10110011

Parent 2: 11001010

We select a crossover point randomly, say after the 4th bit:

Parent 1: 1011 | 0011

Parent 2: 1100 | 1010

After swapping the segments beyond the crossover point, the offspring are:

Child 1: 10111010

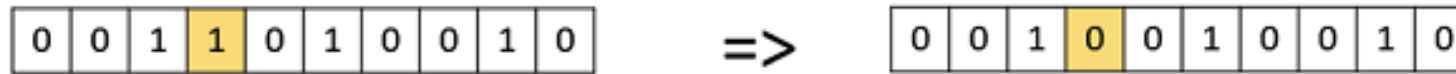
Child 2: 11000011

3. Mutation Operator

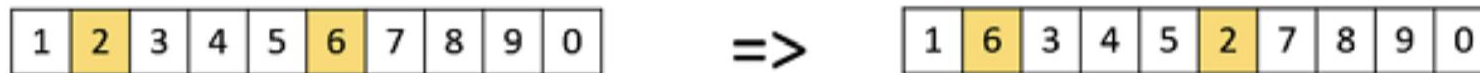
- Mutation is done after Crossover.
- While crossover focuses only on the current solution, the mutation operation searches the whole search space.
- **Mutation is a small, random change in a solution** that helps maintain diversity in the population and find better solutions in genetic algorithms.
- It prevents the algorithm from getting stuck in local optima.
- Though crossover has the main responsibility to search for the optimal solution, mutation is also used for this purpose.

3. Mutation Operator: Types

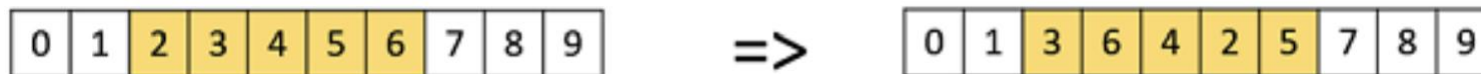
- **Bit Flip Mutation:** In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.



- **Swap Mutation:** In swap mutation, we select two positions on the chromosome at random, and interchange the values.



- **Scramble Mutation:** In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.



Elitism

- Crossover and mutation may destroy the best solution of the population pool
- Elitism is the preservation of few best solutions of the population pool
- Elitism **is a technique** where **the best individuals** (solutions) from the current generation are directly carried over to the next generation without modification.

Nature to Computer Mapping

Nature	Computer
Population	Set of solution
Individual	Solution to a problem
Fitness	Quality of a solution
Chromosome	Encoding for a solution
Gene	Part of the encoding solution

TSP Using GA: Example

- **Cities:** A, B, C, D, E, F
- **Distance Matrix (in km):**

	A	B	C	D	E	F
A	0	4	8	6	7	4
B	4	0	2	5	9	6
C	8	2	0	3	4	2
D	6	5	3	0	6	8
E	7	9	4	6	0	2
F	4	6	2	8	2	0

Objective:

Find the shortest possible route.

TSP Using GA: Example

Step 1: Encoding (Chromosome Representation)

- Each chromosome represents a possible route.


Example

- Chromosome: $A \rightarrow C \rightarrow B \rightarrow E \rightarrow F \rightarrow D \rightarrow A$
- Population consists of multiple random routes.

TSP Using GA: Example

Step 2: Initial Population (Random Routes)

Example Population (5 individuals):

- | | |
|--|--|
| 1) $A \rightarrow F \rightarrow D \rightarrow E \rightarrow C \rightarrow B \rightarrow A$ | Distance: 28 |
| 2) $A \rightarrow C \rightarrow B \rightarrow F \rightarrow E \rightarrow D \rightarrow A$ | Distance: 30 |
| 3) $A \rightarrow D \rightarrow E \rightarrow B \rightarrow C \rightarrow F \rightarrow A$ | Distance: 29 |
| 4) $A \rightarrow E \rightarrow F \rightarrow C \rightarrow B \rightarrow D \rightarrow A$ | Distance: 24  Best |
| 5) $A \rightarrow F \rightarrow C \rightarrow B \rightarrow D \rightarrow E \rightarrow A$ | Distance: 26 |

TSP Using GA: Example

Step 3: Selection Operator (Choosing Parents)

- Select better solutions based on lower total distance.
- **Methods:** Roulette Wheel Selection (Probabilistic method, where probability is proportional to its fitness.)

Example: Selecting two best routes:

- **Parent 1:** $A \rightarrow E \rightarrow F \rightarrow C \rightarrow B \rightarrow D \rightarrow A$ Distance: 26

- **Parent 2:** $A \rightarrow F \rightarrow C \rightarrow B \rightarrow D \rightarrow E \rightarrow A$ Distance: 24

TSP Using GA: Example

Step 4: Crossover (Order Crossover - OX1)

- Creates new solutions (offspring or individuals) by combining parts of two parents.

Parents:

- Parent 1: A → E → F → C → B → D → A
- Parent 2: A → F → C → B → D → E → A

How child 1 created:

- Selected segment from Parent 1:
[E → F → C] (Positions 2 to 4)
- Child 1 gets E → F → C in the same positions.

New solutions

- Child 1: → E → F → C → → →

TSP Using GA: Example

Step 4: Crossover (Order Crossover - OX1)

- Creates new solutions (offspring or individuals) by combining parts of two parents.

Parents:

- Parent 1: A → E → F → C → E → F → A
- Parent 2: A → F → C → B → D → E → A

New solutions

- Child 1: A → E → F → C → B → D → A

How child 1 created:

- Selected segment from Parent 1:
[E → F → C] (Positions 2 to 4)
- Child 1 gets E → F → C in the same positions.
- The remaining positions are filled from Parent 2 while maintaining order and avoiding duplicates.

TSP Using GA: Example

Step 4: Crossover (Order Crossover - OX1)

- Creates new solutions (offspring or individuals) by combining parts of two parents.

Parents:

- Parent 1: A → E → F → C → E → F → A

Same for child 2

- Parent 2: A → F → C → B → D → E → A

New solutions

- Child 1: A → E → F → C → B → D → A

- Child 2: A → F → C → B → D → E → A

TSP Using GA: Example

Step 5: Mutation (Swap Mutation)

- Introduces diversity to avoid premature convergence.
 - **Child 1:** (Swapping Cities E and B):
 - Before: $A \rightarrow E \rightarrow F \rightarrow C \rightarrow B \rightarrow D \rightarrow A$
 - After: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow F \rightarrow E \rightarrow A$
 - **Child 2:** (Swapping Cities C and D):
 - Before: $A \rightarrow F \rightarrow C \rightarrow B \rightarrow D \rightarrow E \rightarrow A$
 - After: $A \rightarrow F \rightarrow D \rightarrow B \rightarrow C \rightarrow E \rightarrow A$

TSP Using GA: Example

Step 5: Final Solution and Conclusion

- After multiple generations, the **best route is:**



A → F → C → B → D → E → A

Distance: 24 km

- GA efficiently finds near-optimal solutions for TSP.
- Works well for larger datasets where brute force is impractical.

Thank You