Solving TSP using Genetic Algorithm

Genetic Algorithms can be used for the TSP problem as follows:

Create a population of candidate TSP solutions. Let the fitness function of the tour be the cost.

⇓

It is a minimization problem.

we will use Path Representation, to start

⇓

In the Path Representation the tour is represented by a permutation of the cities, with the assumption that one returns from the last city in the permutation to the first.

order of the cities visited.

Selection: Clone each tour in proportion to fitness. The cheapest tours are the fittest.

Crossover: Randomly pair the resulting population and perform crossover.

Mutation: Randomly permute a tour once in a while.

TSP :- Single point crossover does not work

P_1 :- | O | D | G | L | A | H | K | M | B | J | F | C | N | I | E |

P_2 :- | H | G | M | F | O | A | D | K | I | C | N | E | L | B | J |

C_1 | O | D | G | L | A | H | K | M | B | J | N | E | L | B | J |

C_2 | H | G | M | F | O | A | D | K | I | C | F | C | N | I | E |

Both offspring are not valid tours.

TSP :- cycle Crossover

1 2 2 3 1 1 2 2 3 3

P_1 :- | O | D | G | L | A | H | K | M | B | J | F | C | N | I | E |

P_2 :- | H | G | M | F | O | A | D | K | I | C | N | E | L | B | J |

Step 1 :- Identify cycles

P_1 :- | O | D | G | L | A | H | K | M | B | J | F | C | N | I | E |

1 2 2 3 1 1 2 2 4 5 3 5 3 4 5

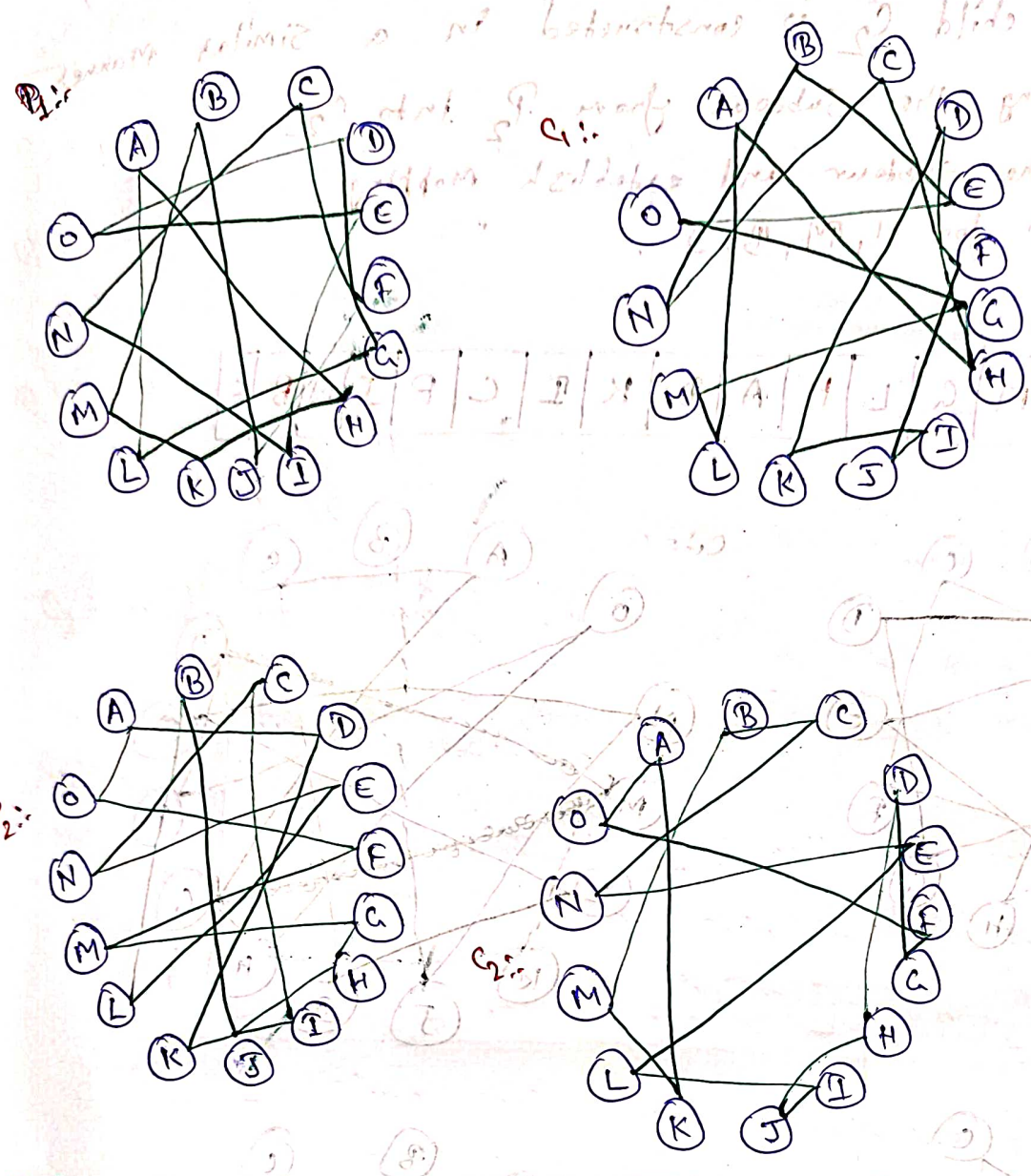
P_2 :- | H | G | M | F | O | A | D | K | I | C | N | E | L | B | J |

Step 2 :-
- C_1 gets even numbered cycles from P_2
- C_1 gets odd numbered cycles from P_1

C_1 :- | O | G | M | L | A | H | D | K | I | J | F | C | N | B | E |

C_2 :-
- C_2 gets odd numbered cycles from P_2
- C_2 gets even numbered cycles from P_1

| H | D | G | F | O | A | K | M | B | C | N | E | L | I | J |



TSP :- Partially Mapped Crossover (PMX).

C1:-

A	G	D	F	O	H	K	M	B	J	N	E	L	I	C

2) (Copy subtours from P_1 into C_1)

P_1:-

O	D	G	L	A	H	K	M	B	J	F	C	N	I	E

1) (Choose a subtour and establish a mapping)

P_2:-

H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

3) would like to copy remaining cities from P_2, but the locations for cities A, D, I, C are occupied by cities H, K, B, J respectively

a) Where should city A be in C_1? follow partial map... H → A

b) Where should be place city D in C_1? K → M → F → D → K (Already in subtour)

c) Similarly do for I and C.

d) Remember that city K is already in C_1.

e) Copy the remaining P_2

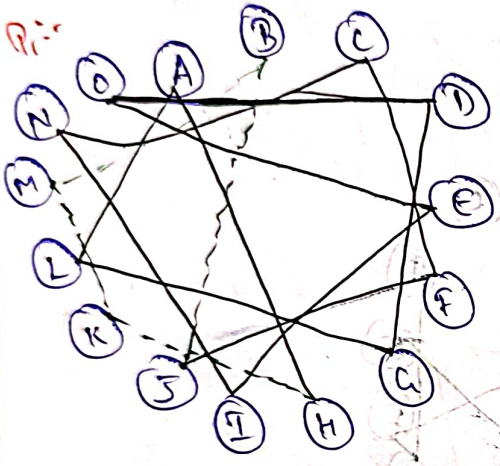
The second child C_2 is constructed in a similar manner.

1) first copying the subtour from P_2 into C_2
2) Choose the subtour and establish mapping.
3) Now, map for H, M, B, J.

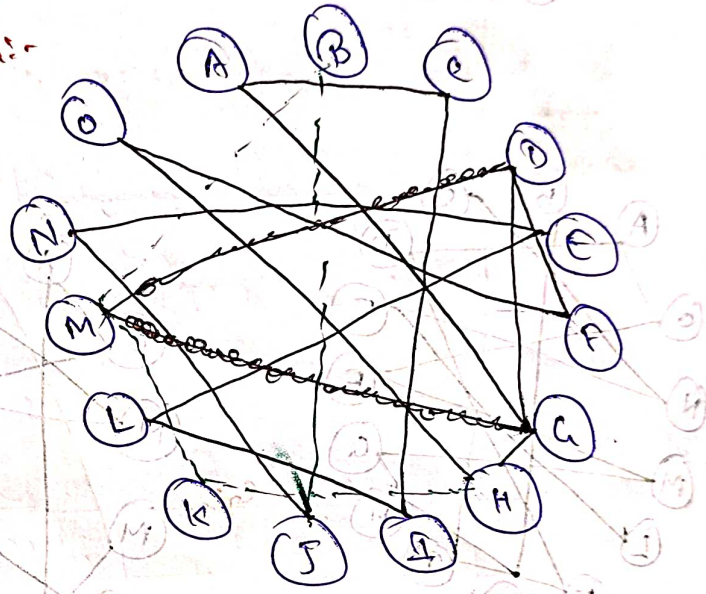
C_2 :-

O	M	G	L	H	A	D	K	I	C	F	J	N	B	E

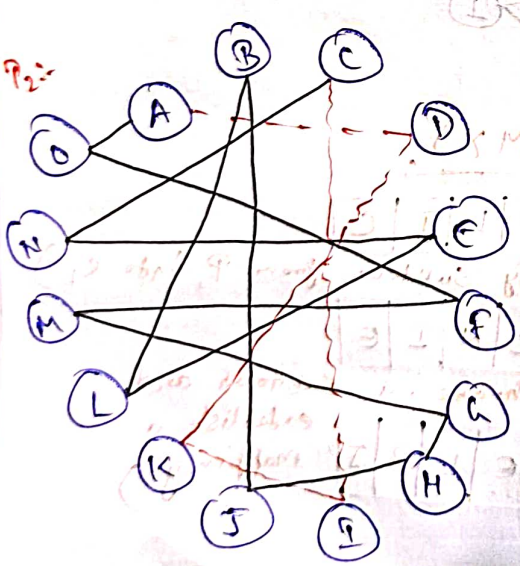
P_1 :-



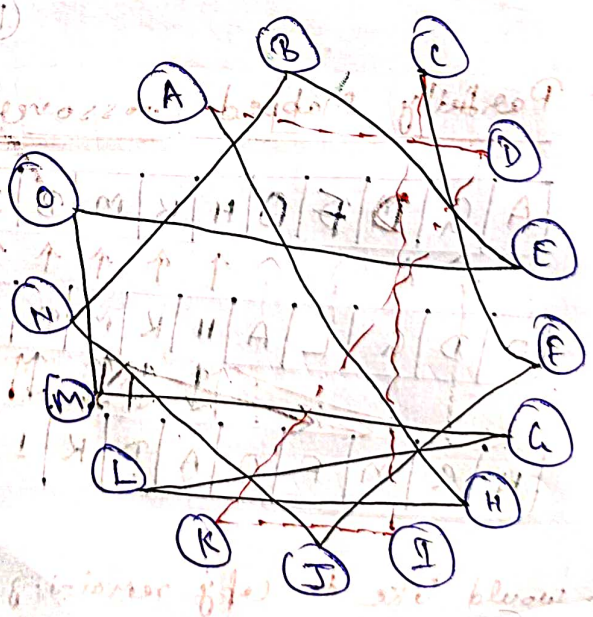
C_1 :-



P_2 :-

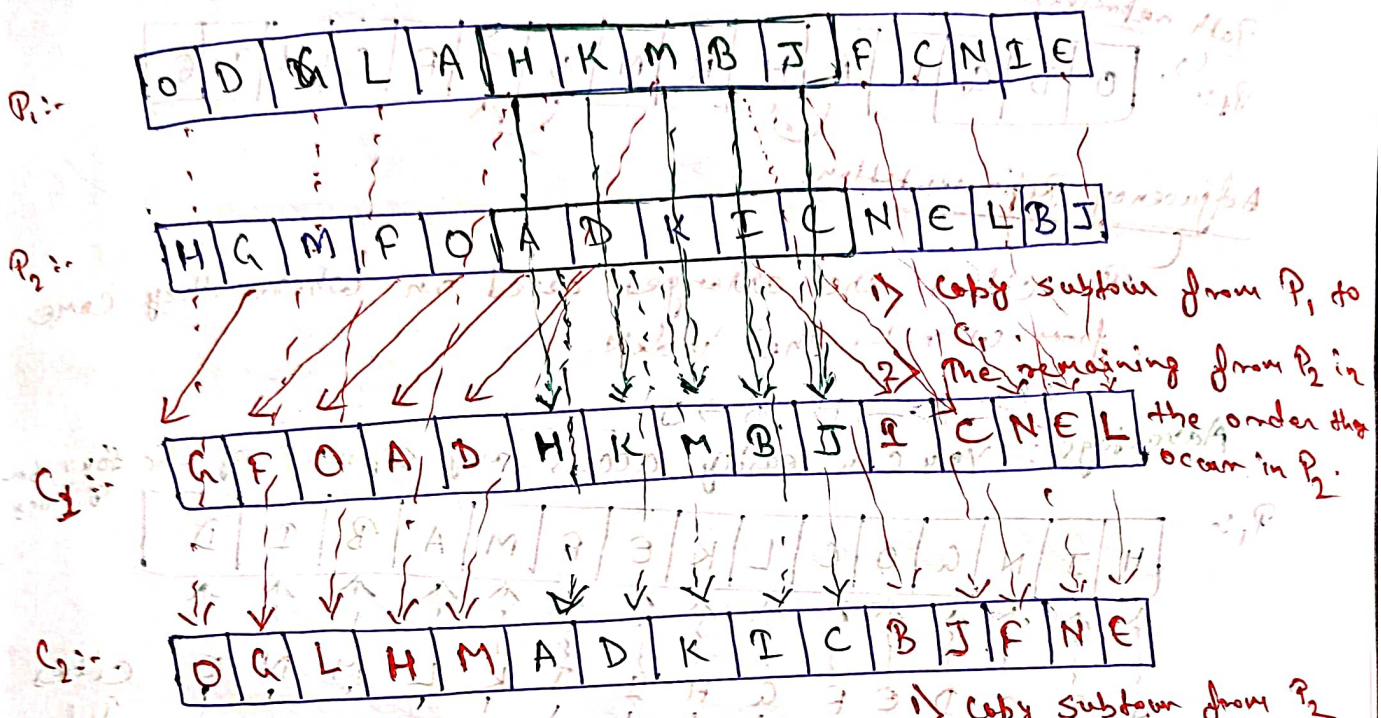


C_2 :-



Note :- Instead of choosing random subtour, we can use heuristic value (Minimum cost) to choose subtour and do the computation.

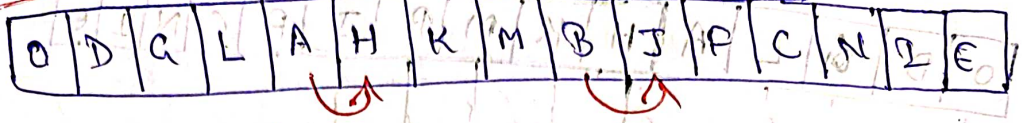
PSP :- Order Crossover.



TSP :- Adjacency Representation

Path representation

P_1 :-



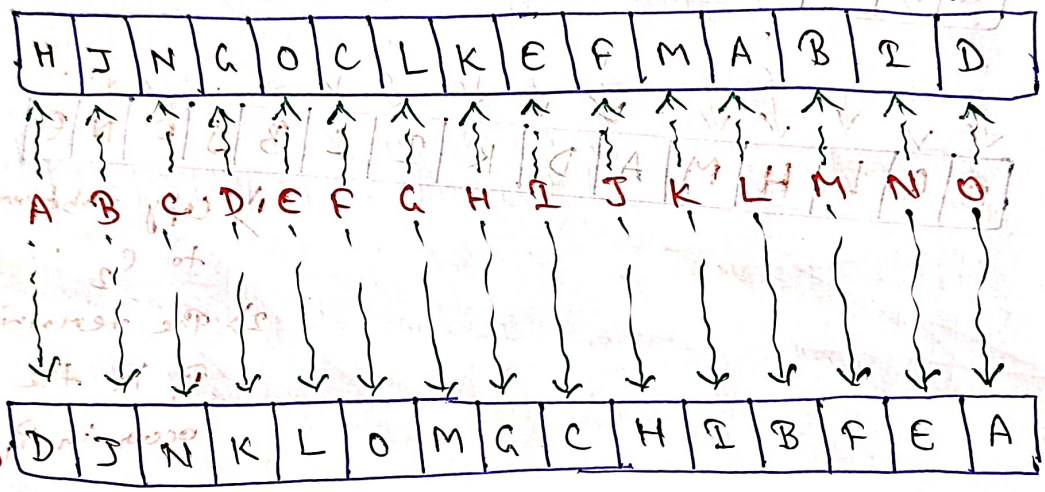
Adjacency Representation

The cities are arranged based on where they come from w.r.t. the 'index'.

Eg :- A → H ; B → J

Advantage :- You can easily access any segment of the tour by index

P_1 :-

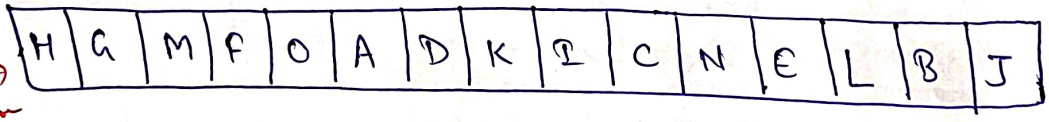


P_2 :-

Adjacency representation

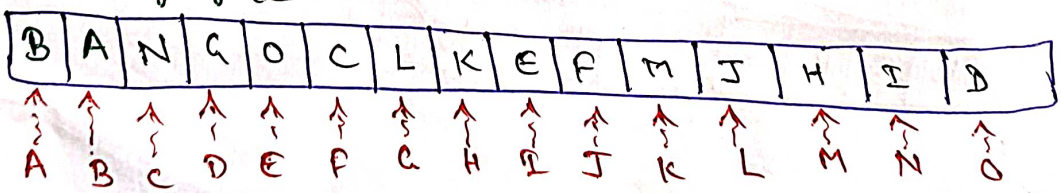
P_2 :-

Path representation



Note :- Suppose example of Adjacency Representation that is not valid because of cycle

P_1 :-



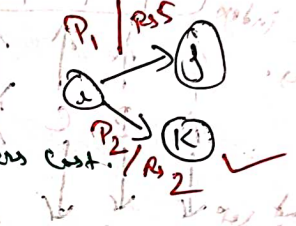
A → B → A

cycle :- Not valid permutation of tour for TSP as candidate Solution.

Adjacency Representation : Crossover Operators

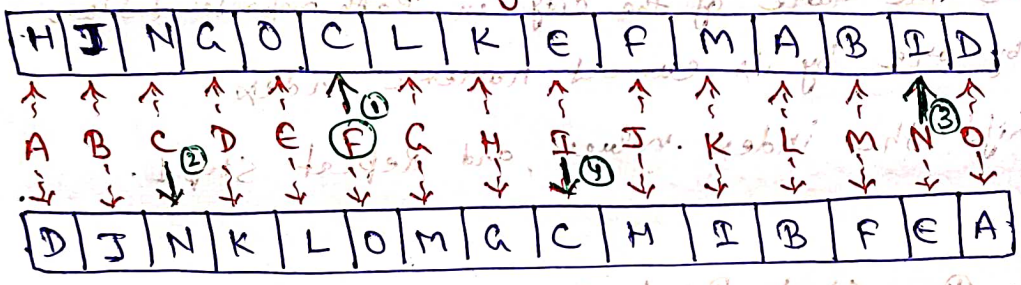
1) Alternating Edges Crossover.
   - Construct a child as follows
   - From a given city A choose the next city B from P_1
   - From the city B choose the next city from P_2
   - Repeat the processing.

2) Heuristic Crossover.
   - For each city choose from that parent (P_1 or P_2) which is closer / less cost.



Note :-

Alternating Edges - careful.



Ex :- Let's say we start with city F

① from P_1 : $F \to C$ :: C

② from P_2 : $C \to N$ :: N

③ from P_1 : $N \to I$ :: I

④ from P_2 : $I \to C$ :: C cycle

Solution :- Choose random next city.