

# Crossover Techniques in GAs

**Debasis Samanta**

Indian Institute of Technology Kharagpur

*dsamanta@iitkgp.ac.in*

11.03.2016

# Important GA Operations

- 1 Encoding
- 2 Fitness Evaluation and Selection
- 3 Mating pool
- 4 Crossover
- 5 Mutation
- 6 Inversion
- 7 Convergence test

# Important GA Operations

- 1 Encoding
- 2 Fitness evaluation and Selection
- 3 Mating pool
- 4 **Crossover**
- 5 Mutation
- 6 Inversion
- 7 Convergence test

# Reproduction in Genetic Algorithm

## Reproduction:

- **Crossover**
- Mutation
- Inversion

These genetic operators varies from one encoding scheme to another.

- Binary coded GAs
- Real-coded GAs
- Tree-coded GAs

# Mating Pool: Prior to crossover operation

- A mating pair (each pair consists of two strings) are selected at random. Thus, if the size of mating pool is  $N$ , then  $\frac{N}{2}$  mating pairs are formed. [Random Mating]
- The pairs are checked, whether they will participate in reproduction or not by tossing a coin, whose probability being  $p_c$ . If  $p_c$  is head, then the parent will participate in reproduction. Otherwise, they will remain intact in the population.

## Note :

Generally,  $p_c = 1.0$ , so that almost all the parents can participate in production.

# Crossover operation

Once, a pool of mating pair are selected, they undergo through crossover operations.

- 1 In crossover, there is an exchange of properties between two parents and as a result of which **two** offspring solutions are produced.
- 2 The crossover point(s) (also called k-point(s)) **is(are)** decided using a random number generator generating integer(s) in between 1 and  $L$ , where  $L$  is the length of the chromosome.
- 3 Then we perform exchange of gene values with respect to the k-point(s)

There are many exchange mechanisms and hence crossover strategies.

# Crossover Techniques in Binary Coded GA

# Crossover operations in Binary-coded GAs

- There exists a large number of crossover schemes, few important of them are listed in the following.
  - 1 Single point crossover
  - 2 Two-point crossover
  - 3 Multi-point crossover (also called n-point crossover)
  - 4 Uniform crossover (UX)
  - 5 Half-uniform crossover (HUX)
  - 6 Shuffle crossover
  - 7 Matrix crossover (Two-dimensional crossover)
  - 8 Three parent crossover



# Single point crossover

- 1 Here, we select the  $K$ -point lying between 1 and  $L$ . Let it be  $k$ .
- 2 A single crossover point at  $k$  on both parent's strings is selected.
- 3 All data **beyond that point** in either string is swapped between the two parents.
- 4 The resulting strings are **the chromosomes of the offsprings produced**.

# Single point crossover: Illustration

Before Crossover

Parent 1 :

0	1	1	0	0	0	1	0
---	---	---	---	---	---	---	---

Parent 2 :

1	0	1	0	1	1	0	0
---	---	---	---	---	---	---	---

Two diploid  
from a  
mating pair

Crossover Point - k

Select crossover points randomly

Offspring 1:

0	1	1	0	1	1	0	0
---	---	---	---	---	---	---	---

Offspring 2:

1	0	1	0	0	0	1	0
---	---	---	---	---	---	---	---

Two diploid  
for two new  
offspring is  
produced

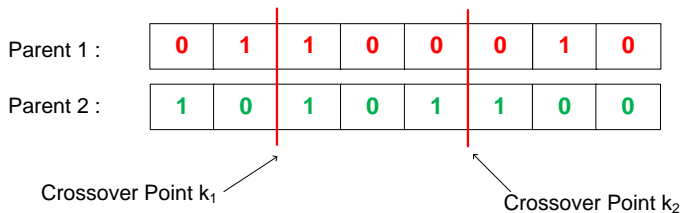
After Crossover

# Two-point crossover

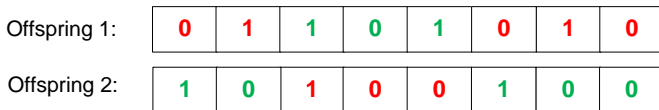
- 1 In this scheme, we select two different crossover points  $k_1$  and  $k_2$  lying between 1 and  $L$  at random such that  $k_1 \neq k_2$ .
- 2 The middle parts are swapped between the two strings.
- 3 Alternatively, left and right parts also can be swapped.

# Two-point crossover: Illustration

Before Crossover



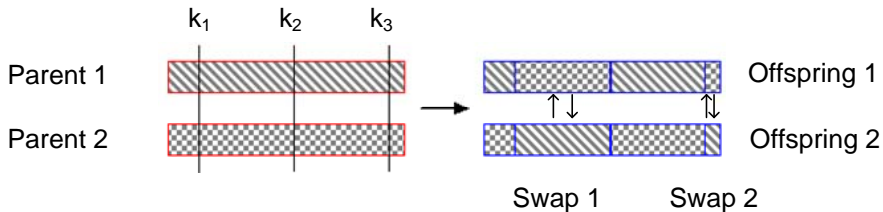
Select two crossover points randomly



After Crossover

# Multi-point crossover

- 1 In case of multi-point crossover, a number of crossover points are selected along the length of the string, at random.
- 2 The bits lying between alternate pairs of sites are then swapped.



# Uniform Crossover (UX)

- Uniform crossover is a more general version of the multi-point crossover.
- In this scheme, at each bit position of the parent string, we toss a coin (with a certain probability  $p_s$ ) to determine whether there will be swap of the bits or not.
- The two bits are then swapped or remain unaltered, accordingly.

# Uniform crossover (UX): Illustration

Before crossover

Parent 1 :

1	1	0	0	0	1	0	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Parent 2 :

0	1	1	0	0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Coin tossing:

1	0	0	1	1	1	0	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---

After crossover

Offspring 1:

1	1	1	0	0	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Offspring 2:

0	1	0	0	0	1	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Rule: If the toss is 0 then swap the bits between P1 and P2

# Uniform crossover with crossover mask

- Here, each gene is created in the offspring by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosome.
- Where there is a 1 in the mask, the gene is copied from the first parent
- Where there is a 0 in the mask, the gene is copied from the second parent.
- The reverse is followed to create another offsprings.



# Uniform crossover with crossover mask: Illustration

Before Crossover

Parent 1 :

1	1	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Parent 2 :

0	1	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Mask

1	0	0	1	1	1	0	1
---	---	---	---	---	---	---	---

Offspring 1:

1	1	1	0	0	1	1	1
---	---	---	---	---	---	---	---

When there is a 1 in the mask, the gene is copied from Parent 1 else from Parent 2.

Offspring 2:

0	1	0	0	0	1	0	1
---	---	---	---	---	---	---	---

When there is a 1 in the mask, the gene is copied from Parent 2 else from Parent 1.

After Crossover

# Half-uniform crossover (HUX)

- In the half uniform crossover scheme, exactly half of the non-matching bits are swapped.
  - 1 Calculate the Hamming distance (the number of differing bits) between the given parents.
  - 2 This number is then divided by two.
  - 3 The resulting number is how many of the bits that do not match between the two parents will be swapped but probabilistically.
  - 4 Choose the locations of these half numbers (with some strategies, say coin tossing) and swap them.

# Half-uniform crossover: Illustration

Before crossover

Parent 1 :

1	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

Here, Hamming distance is 4

Parent 2 :

1	0	0	1	1	0	1	1
---	---	---	---	---	---	---	---

Tossing:

	1		0	1			1
--	---	--	---	---	--	--	---

If toss is 1, then swap the bits else remain as it is

Offspring 1:

1	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

Offspring 2:

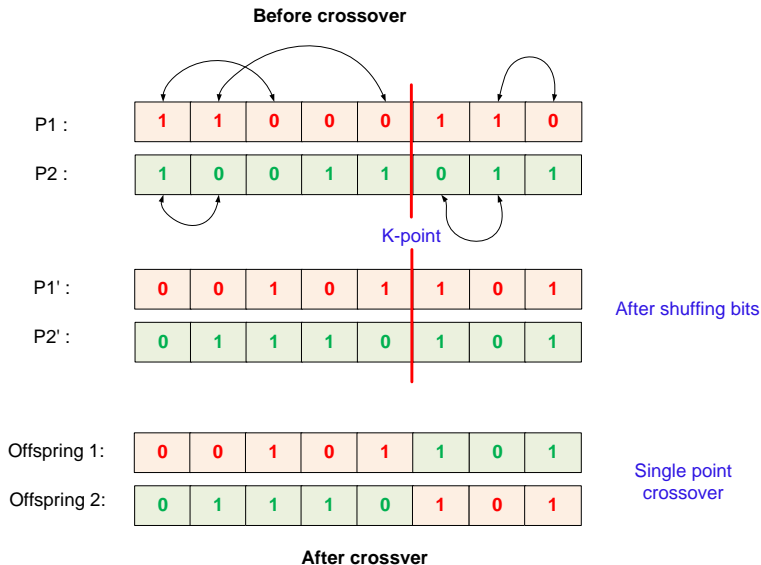
1	1	0	1	0	0	1	0
---	---	---	---	---	---	---	---

After crossover

# Shuffle crossover

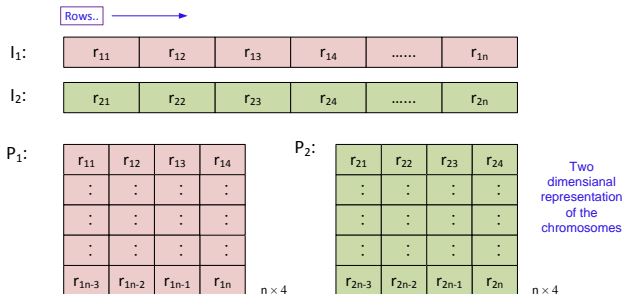
- A single crossover point is selected. It divides a chromosome into two parts called schema.
- In both parents, genes are shuffled in each schema. Follow some strategy for shuffling bits
- Schemas are exchanged to create offspring (as in single crossover)

# Shuffle crossover: Illustration

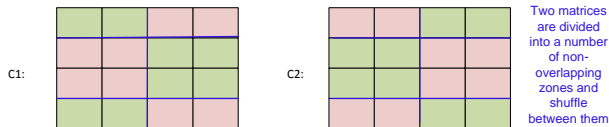


# Matrix crossover

The matrix crossover strategy is explained with the following illustration.



Then matrices are divided into a number of non-overlapping zones



# Three parent crossover

- In this techniques, three parents are randomly chosen.
- Each bit of the first parent is compared with the bit of the second parent.
- If both are the same, the bit is taken for the offspring.
- Otherwise, the bit from the third parent is taken for the offspring.

# Three parent crossover: Illustration

P1:	1	1	0	1	0	0	0	1
P2:	0	1	1	0	1	0	0	1
P3:	0	1	1	0	1	1	0	1
C1:	0	1	1	0	1	0	0	1

**Note:** Sometime, the third parent can be taken as the crossover mask.



# Comments on the binary crossover techniques

## 1 Non-uniform variation:

It can not combine all possible schemas (i.e. building blocks)

For example : it can not in general combine instances of

1 1 \* \* \* \* 1

and

\* \* \* \* 1 1 \* \*

to form an instance of

1 1 \* \* 1 1 \* 1.

## 2 Positional bias:

The schemas that can be created or destroyed by a crossover depends strongly on the location of the bits in the chromosomes.

# Comments on the binary crossover techniques

## 3 End-point bias:

It is also observed that single-point crossover treats some loci preferentially, that is, the segments exchanged between the two parents always contain the end points of the strings.

## 4 Hamming cliff problem:

A one-bit change can make a large (or a small) jump.

A multi-bits can make a small (or a large gap).

For example, **1000**  $\Rightarrow$  **0111**

(Here, Hamming distance = 4, but distance between phenotype is 1)

Similarly, **0000**  $\Rightarrow$  **1000**

(Here, Hamming distance = 1, but distance between phenotype is 8)

# Comments on the binary crossover techniques

- To reduce the positional bias and end-point bias, two-point crossover and multi-point crossover schemes have been evolved.
- In contrast, UX and HUX distribute the patterns in parent chromosomes largely resulting too much deflections in the offspring.
- To avoid binary code related problem, **gray coding** can be used.
- In summary, binary coding is the simplest encoding and its crossover techniques are fastest compared to the crossover techniques in other GA encoding schemes.

# Crossover Techniques in Real Coded GA

# Crossover techniques in Real coded GA

Following are the few well known crossover techniques for the real-coded GAs.

- Linear crossover
- Blend crossover
- Binary simulated crossover

# Linear crossover in Real-coded GAs

- This scheme uses some linear functions of the parent chromosomes to produce the new children.

- **For example**

Suppose  $P_1$  and  $P_2$  are the two parameter's values in two parents, then the corresponding offspring values in chromosomes can be obtained as

$$C_i = \alpha_i P_1 + \beta_i P_2$$

where  $i = 1, 2 \dots n$  (number of children).

$\alpha_i$  and  $\beta_i$  are some constants.

# Linear crossover: An example

- Example :

Suppose  $P_1 = 15.65$  and  $P_2 = 18.83$

$$\alpha_1 = 0.5 = \beta_1$$

$$\alpha_2 = 1.5 \text{ and } \beta_2 = -0.5$$

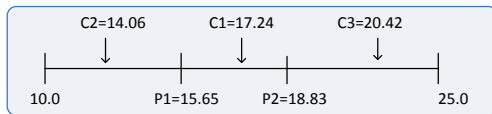
$$\alpha_3 = -0.5 \text{ and } \beta_3 = 1.5$$

- Answer :

$$C_1 = 0.5 \times (P_1 + P_2) = 17.24$$

$$C_2 = 1.5 \times P_1 - 0.5 \times P_2 = 14.06$$

$$C_3 = -0.5 \times P_1 + 1.5 \times P_2 = 20.24$$



# Advantages and limitations

## Advantages

- 1 It is simple to calculate and hence faster in computation
- 2 Can allow to generate a large set of offspring from two parent values
- 3 Controls are possible to choose a wide-range of variations

## Limitations

- 1 Needs to be decided the values of  $\alpha_i$  and  $\beta_i$
- 2 It is difficult for the inexperienced users to decide the right values for  $\alpha_i$  and  $\beta_i$
- 3 If  $\alpha_i$  and  $\beta_i$  values are not chosen properly, the solution may stuck into a local optima.



# Blend crossover in Real-coded GAs

This scheme can be stated as follows.

- 1 Let  $P_1$  and  $P_2$  are the two parameter's values in two parent's chromosomes, such that  $P_1 < P_2$
- 2 Then the blend crossover scheme creates the children solution lying in the range

$$\langle \{P_1 - \alpha(P_2 - P_1)\} \cdots \{P_2 - \alpha(P_2 - P_1)\} \rangle$$

where  $\alpha$  is a constant to be decided so that children solution do not come out of the range of domain of the said parameter.

# Blend crossover in Real-coded GAs

- 3 Another parameter  $\gamma$  has to be identified by utilizing the  $\alpha$  and a random number  $r$  in the range of (0.0, 1.0) both exclusive like the following:

$$\gamma = (1 + 2\alpha) r - \alpha$$

- 4 The children solutions  $C_1$  and  $C_2$  are determined from the parents as follows,

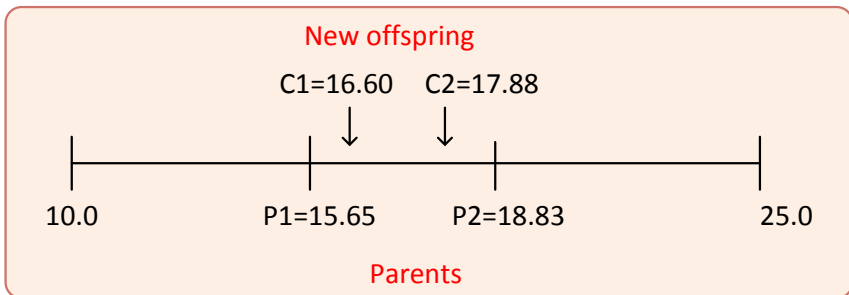
$$\begin{aligned} C_1 &= (1 - \gamma) P_1 + \gamma P_2 \\ C_2 &= (1 - \gamma) P_2 + \gamma P_1 \end{aligned}$$

# Blend crossover : An example

- Example :

$$P_1 = 15.65 \text{ and } P_2 = 18.83$$

$$\alpha = 0.5 \text{ and } \gamma = 0.6$$



# Simulated binary crossover in Real-coded GAs

- This scheme is based on the probability distribution of generated children solution from the given parents.
- A spread factor  $\alpha$  is used to represent the spread of the children solutions with respect to that of the parents, as given below.

$$\alpha = \left| \frac{C_1 - C_2}{P_1 - P_2} \right|$$

Here  $P_1$  and  $P_2$  are represent the parent points and  $C_1$  and  $C_2$  are two children solutions.

# Simulated binary crossover in Real-coded GAs

Three different cases may occurs:

- Case 1:  $\alpha < 1$  (Contracting Crossover)

The spread of children is less than the parents.

- Case 2:  $\alpha > 1$  (Expanding Crossover)

The spread of children is more than the parents.

- Case 3:  $\alpha = 1$  (Stationary Crossover)

The spread of children is same as that of the parents.

# Simulated Binary Crossover

## Probability Distribution:

- Case 1: For Contracting Crossover

$$C(\alpha) = 0.5(q+1)\alpha^2$$

- Case 2: For Expanding Crossover

$$E(\alpha) = 0.5(q+1)\frac{1}{\alpha^{q+2}}$$

# Simulated binary crossover in Real-coded GAs

Following steps are used to create two children solutions  $C_1$  and  $C_2$  from the parents  $P_1$  and  $P_2$ .

1 Create a random number  $r \in \{0.0 \dots 1.0\}$

2 Determine  $\alpha'$  such that

$$\int_0^{\alpha'} C(\alpha) d\alpha = r, \text{ if } r < 0.5$$

and

$$\int_1^{\alpha'} E(\alpha) d\alpha = r, \text{ if } r > 0.5$$

3 Using the value of  $\alpha'$  obtain two children solution as follows

- $C_1 = 0.5 [(P_1 + P_2) - \alpha' |P_2 - P_1|]$
- $C_2 = 0.5 [(P_1 + P_2) + \alpha' |P_2 - P_1|]$

# Simulated binary crossover in Real-coded GAs

**Example:**

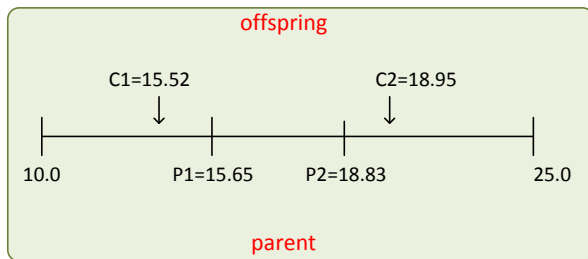
$$P_1 = 15.65$$

$$P_2 = 18.83$$

$$q = 2$$

$$\alpha' = 1.0772$$

Assuming expanding crossover with  $r > 0.5$





# Advantages and limitations

## Advantages

- 1 We can generate a large number of offspring from two parents.
- 2 More explorations with diverse offspring.
- 3 Results are accurate and usually terminated with global optima.
- 4 Termination with a less number of iterations.
- 5 Crossover techniques are independent of the length of the chromosome.

## Limitations

- 1 Computationally expensive compared to binary crossover.
- 2 If proper values of parameters involved in the crossover techniques are not chosen judiciously, then it may lead to premature convergence with not necessarily optimum solutions.

# Crossover Techniques in Order GAs

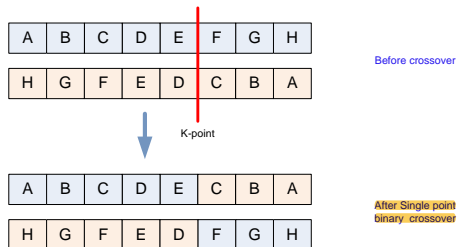
# Crossover techniques in order GA

- Any binary crossover techniques are not applicable to Order coded GAs

## Example

**Reference: TSP**

Consider any two chromosomes with Order-coded encoding scheme



Here, the offspring are not valid chromosomes

- Since, sequence of gene values are important, Real-coded crossover techniques, which are to produce real number from two given real numbers are also not applicable to Order-coded GAs.

# Crossover techniques in order GA

Some important crossover techniques in Order-coded GAs are:

- 1 Single-point order crossover
- 2 Two-point order crossover
- 3 Partially mapped crossover (PMX)
- 4 Position based crossover
- 5 Precedence-preservation crossover (PPX)
- 6 Edge recombination crossover

**Assumptions:** For all crossover techniques, we assume the following:

- Let  $L$  be the length of the chromosome.
- $P_1$  and  $P_2$  are two parents (are selected from the mating pool).
- $C_1$  and  $C_2$  denote offspring (initially empty).

# Single point order crossover

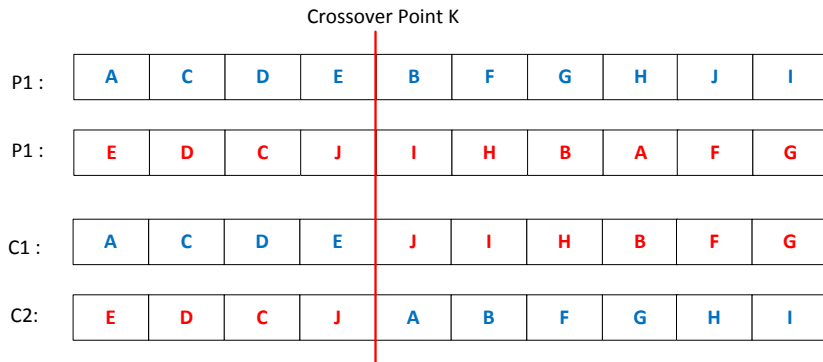
Given two parents  $P_1$  and  $P_2$  with chromosome length, say  $L$ .

## Steps:

- 1 Randomly generate a crossover point  $K$  such that  $(1 < K < L)$ .
- 2 Copy the left schema of  $P_1$  into  $C_1$  (initially empty) and left schema of  $P_2$  into  $C_2$  (also initially empty).
- 3 For the schema in the right side of  $C_1$ , copy the gene value from  $P_2$  in the same order as they appear but not already present in the left schema.
- 4 Repeat the same procedure to complete  $C_2$  from  $P_1$ .

# Single point order crossover: Illustration

## Example :



# Two-point order crossover

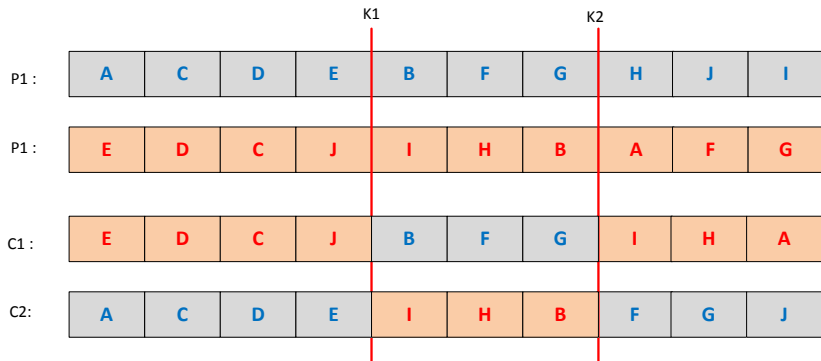
It is similar to the single-point order crossover, but with two  $k$ -points.

## Steps:

- 1 Randomly generate two crossover points  $K_1$  and  $K_2$ .  $1 < K_1, K_2 < L$
- 2 The schema in middle of  $P_1$  and  $P_2$  are copied into  $C_1$  and  $C_2$  (initially both are empty), respectively in the same location as well as in the same order.
- 3 The remaining schema in  $C_1$  and  $C_2$  are copied from  $P_2$  and  $P_1$  respectively, so that an element already selected in child solution does not appear again.

# Two-point order crossover: Illustration

## Example :





# Precedence preservation order crossover

Let the parent chromosomes be  $P_1$  and  $P_2$  and the length of chromosomes be  $L$ .

## Steps:

- (a) Create a vector  $V$  of length  $L$  randomly filled with elements from the set  $\{1, 2\}$ .
- (b) This vector defines the order in which genes are successfully drawn from  $P_1$  and  $P_2$  as follows.
  - ① We scan the vector  $V$  from left to right.
  - ② Let the current position in the vector  $V$  be  $i$  (where  $i = 1, 2, \dots, L$ ).
  - ③ Let  $j$  (where  $j = 1, 2, \dots, L$ ) and  $k$  (where  $k = 1, 2, \dots, L$ ) denotes the  $j^{th}$  and  $k^{th}$  gene of  $P_1$  and  $P_2$ , respectively. Initially  $j = k = 1$ .

# Precedence preservation order crossover

- 4 If  $j^{th}$  value is 1 then

Delete  $j^{th}$  gene value from  $P_1$  and as well as from  $P_2$  and append it to the offspring (which is initially empty).

- 5 Else

Delete  $k^{th}$  gene value from  $P_2$  and as well as from  $P_1$  and append it to the offspring.

- 6 Repeat Step 2 until both  $P_1$  and  $P_2$  are empty and the offspring contains all gene values.

# Precedence preservation order crossover : Example

## Example :

Random Vector  $\sigma$

2	1	1	2	1	1	2	2	1	2
---	---	---	---	---	---	---	---	---	---

P1 :

A	C	D	E	B	F	G	H	J	I
---	---	---	---	---	---	---	---	---	---

P2 :

E	D	C	J	I	H	B	A	F	G
---	---	---	---	---	---	---	---	---	---

C1 :

E	C	D	J	B	F	H	A	I	G
---	---	---	---	---	---	---	---	---	---

C2:

?

**Note :** We can create another offspring following the alternative rule for 1 and 2.

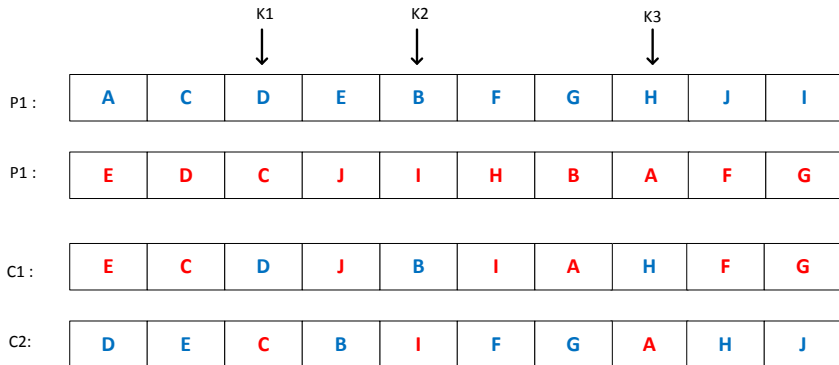
# Position-based order crossover

## Steps :

- 1 Choose  $n$  crossover points  $K_1, K_2 \dots K_n$  such that  $n \ll L$ , the length of chromosome.
- 2 The gene values at  $K_1^{th}, K_2^{th} \dots K_n^{th}$  positions in  $P_1$  are directly copied into offspring  $C_1$  (Keeping their position information intact).
- 3 The remaining gene values in  $C_1$  will be obtained from  $P_2$  in the same order as they appear there except they are already not copied from  $P_1$ .
- 4 We can reverse the role of  $P_1$  and  $P_2$  to get another offspring  $C_2$ .

# Position-based order crossover : Example

Let us consider three  $k$ -points namely  $K_1$ ,  $K_2$  and  $K_3$  in this example.



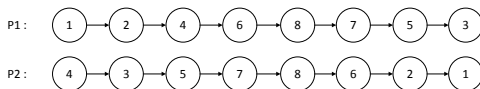
# Edge recombination order crossover

- This crossover technique is used to solve TSP problem when the cities are not completely connected to each other.
- In this technique, an edge table which contains the adjacency information (but not the order).
- In the other words, edge table provides connectivity information.

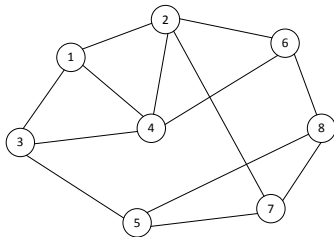
# Edge recombination order crossover: Illustration

## Example

- Let us consider a problem instance of a TSP with 9 cities.
- Assume any two chromosome  $P_1$  and  $P_2$  for the mating.

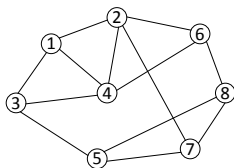


## Connectivity graph:



# Edge recombination order crossover: Illustration

## Edge table for the connectivity graph:



City	Connectivity
1	2 4 3
2	1 4 7 6
3	1 4 5
4	1 2 3 6
5	3 7 8
6	2 4 8
7	2 5 8
8	5 6 7



# Edge recombination order crossover: Illustration

## Steps:

Let the child chromosome be  $C_1$  (initially empty).

- 1 Start the child tour with the starting city of  $P_1$ . Let this city be  $X$ .
- 2 Append city  $X$  to  $C$ .
- 3 Delete all occurrences of  $X$  from the connectivity list of all cities (right-hand column).
- 4 From city  $X$  choose the next city say  $Y$ , which is in the list of minimum (or any one, if there is no choice) connectivity links.
- 5 Make  $X = Y$  [ i.e. new city  $Y$  becomes city  $X$ ].
- 6 Repeat Steps 2-5 until the tour is complete.
- 7 End

# Any questions??