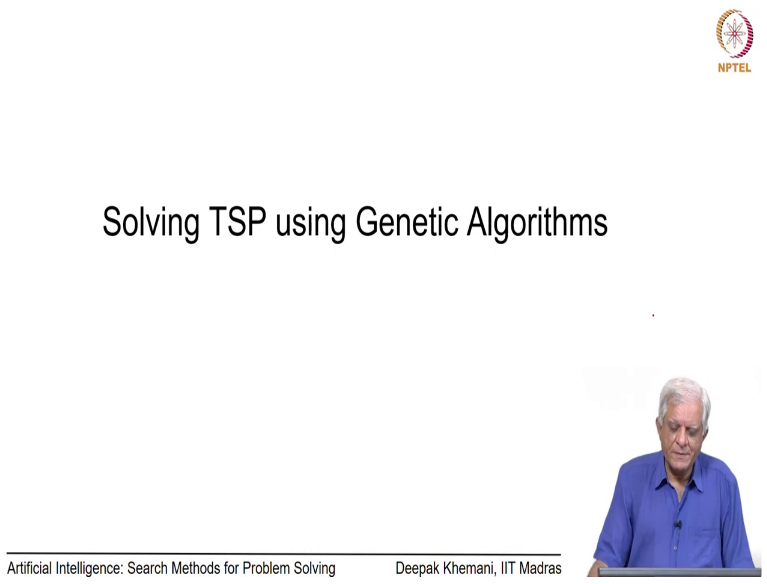
Artificial Intelligence: Search Methods for Problem Solving

Prof. Deepak Khemani
Department of Computer Science & Engineering
Indian Institute of Technology, Madras

Chapter - 04
A First Course in Artificial Intelligence
Lecture – 31
Population Based Methods: Genetic Algorithms for the TSP

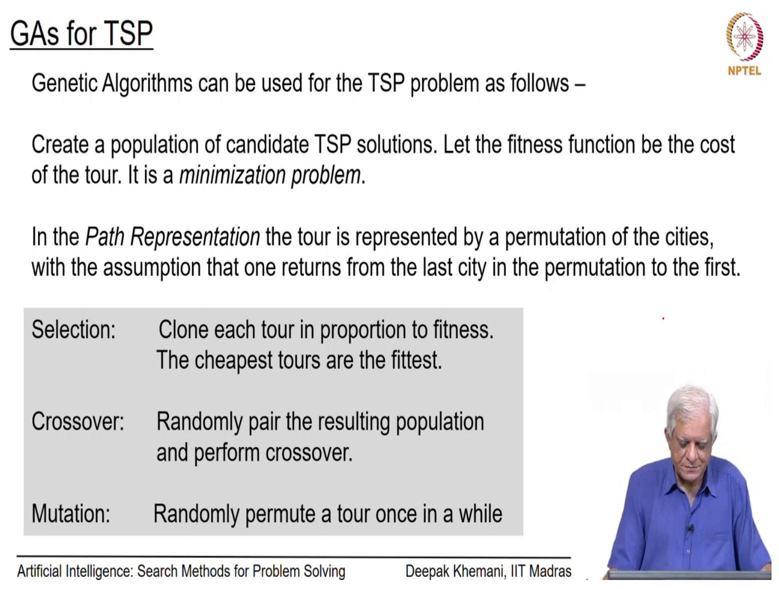
(Refer Slide Time: 00:14)



Solving TSP using Genetic Algorithms

So, welcome back. In the last session, we saw how GS can solve problems like SAT for example, and we saw a tiny problem tiniest tiny problem of optimization. So, let us now focus on TSP. As we have been saying repeatedly, SAT and TSP are two kind of classical problems in computing which we will need to solve not only because they are hard in nature, but also because they have many applications in the real world essentially.

(Refer Slide Time: 00:49)



GAs for TSP

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

Create a population of candidate TSP solutions. Let the fitness function be the cost of the tour. It is a *minimization problem*.

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.

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

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

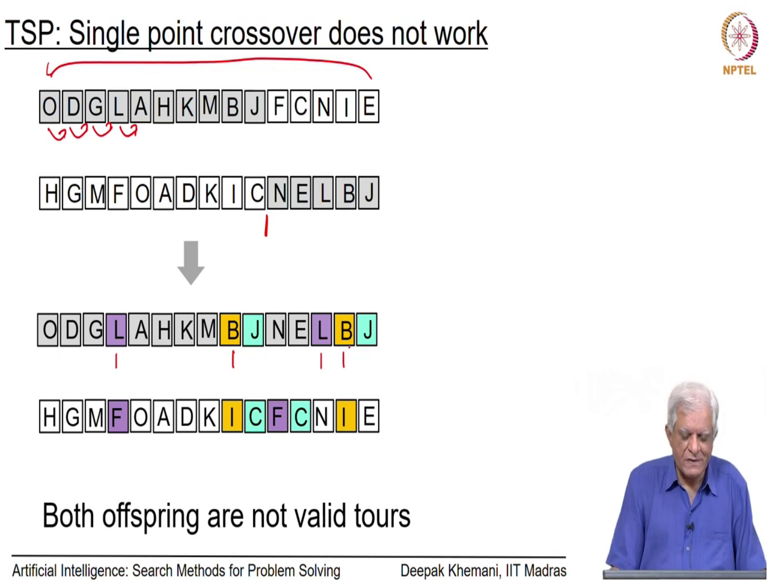
So, let us focus on TSP in this session and we want to you see how genetic algorithms can be use for TSP. So, we start off by creating a population of candidate TSP solutions and the fitness function will be the cost of the tour essentially.

Now observe that since we want the minimum cost tour, the fitness function this has to be kind of considered appropriately and this is a minimization problem. So, one thing you could do is you could take the cost of the tour and subtract it from some large constant and the difference would be a maximization problem, but that is besides the point. As long as you take care that the fitter candidates get greater chance for selection, the algorithm should work fine.

So, we will use a path representation of the tour to start with and in this path representation, the tour is represented by a permutation of the cities and the permutation simply says that you are going the order in which you are going to visit the cities in the tour. And the assumption is that the last city in the permutation from the last city in the permutation you move to the first city back because it has to be a complete cycle, but you mentioned each city exactly one. So, every permutation is a tour essentially.

So, we go through this process of genetic algorithm. Clone each tour in proportion to fitness. Remember that the cheapest tours are the fittest. Crossover randomly pair the resulting population and perform crossover and once in a while randomly permute the tour and that would serve as a mutation essentially.

(Refer Slide Time: 02:39)



TSP: Single point crossover does not work

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

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

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

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

Both offspring are not valid tours

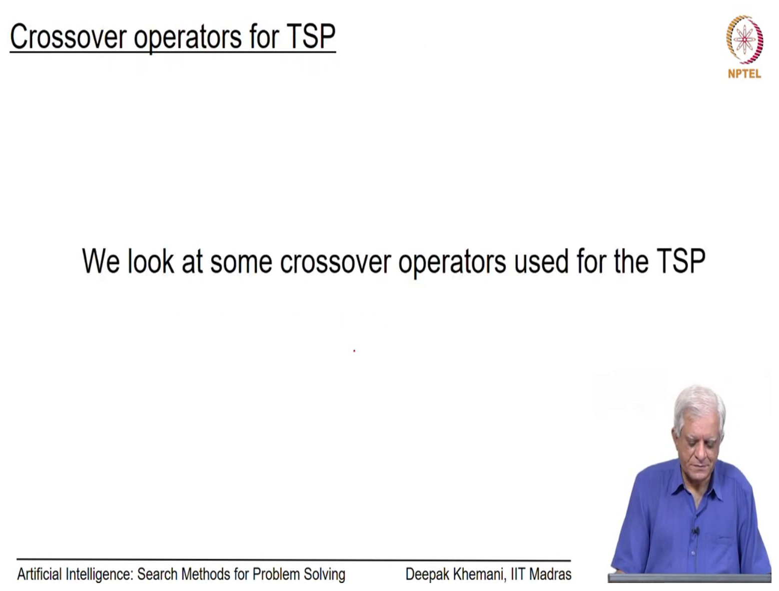
Artificial Intelligence: Search Methods for Problem Solving — Deepak Khemani, IIT Madras

Now, does what if we tried to do single point crossover essentially? So, if we wanted to do single point crossover and here are two example tours. In the first tour, you start with city O go to city D, then to G, then to L and so on and so that is the path representation you are seeing from where and from the last city, you will come back to O and that is a complete tour.

And supposing we do crossover across this point. So, we have already shaded the cities so that the final cities are seen in grey, one child is seen in grey and the other child is seen in white. But, if you look at both of the children which are given below, then in both the children there is repetition of cities happening. So, L is being repeated here, B is being repeated here. Now obviously, this is not a valid tour essentially.

So, this simply means that single point crossover is not suitable. At least, it is not suitable for the path representation. Maybe we will later on see that there is an alternate representation where we can in fact, use the single point crossover. So, let us look at some other crossovers that are popular in literature for solving the TSP ok.

(Refer Slide Time: 03:53)

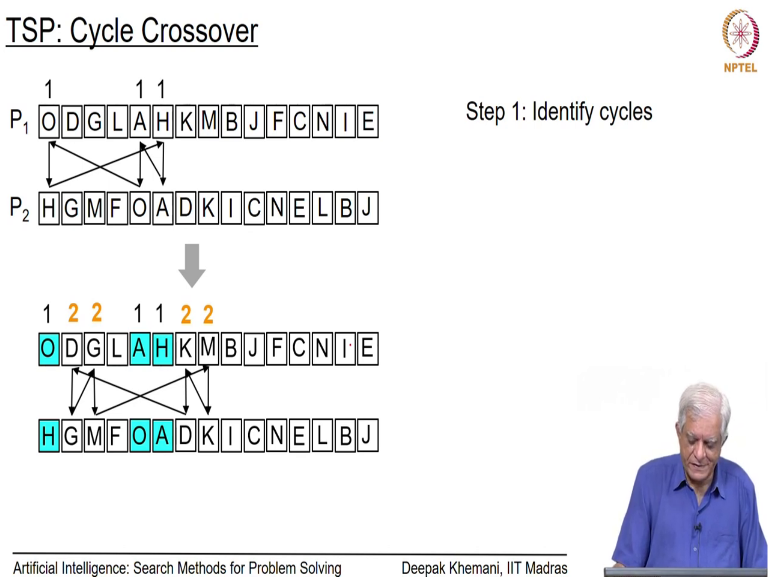


Crossover operators for TSP

We look at some crossover operators used for the TSP

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

(Refer Slide Time: 03:56)



TSP: Cycle Crossover

Step 1: Identify cycles

1 1 1
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
1 2 2 1 1 2 2
O D G L A H K M B J F C N I E
H G M F O A D K I C N E L B J

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

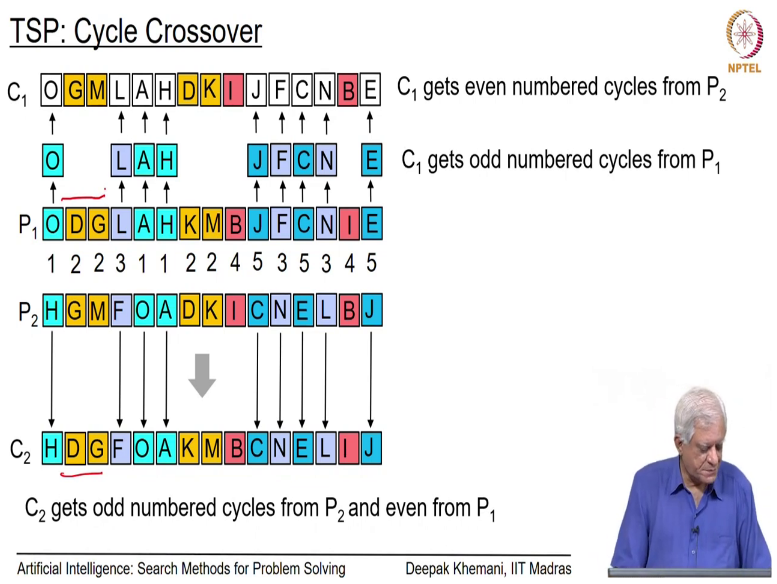
So, here is something which is called a cycle crossover and the process works like this; that given these two parents which are the same parents that we looked at in the for single point crossover.

You first identify cycles and the cycles identified by a process where you say for example, that you start with the city O and you look at what is the city which is in the corresponding position in the second parent and we can see that, that is a city H.

So, we have started constructing the first cycle, then you look at where is H in the second parent and we or where A which is a corresponding city for A is in the second parent, then for correspond for A the corresponding cities O and they that takes us back to the original O and so we have identified one cycle in these two parents and we have shown the cycle by covering it in this sign color.

We repeat this process. We start with city D now the second city in parent 1 and look for the second cycle. So, the same process follows that we follow where the corresponding cities in the first parent and we keep doing this till we have reached the original city back. So, here we started with D went to G, from G we went to M, from M we went to K and from K we came back to D and this gives us the second cycle. So, we continue this process.

(Refer Slide Time: 05:44)



TSP: Cycle Crossover

C_1 O G M L A H D K I J F C N B E
O L A H J F C N E
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
C_2 H D G F O A K M B C N E L I J
NPTEL

C_1 gets even numbered cycles from P_2

C_1 gets odd numbered cycles from P_1

C_2 gets odd numbered cycles from P_2 and even from P_1

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

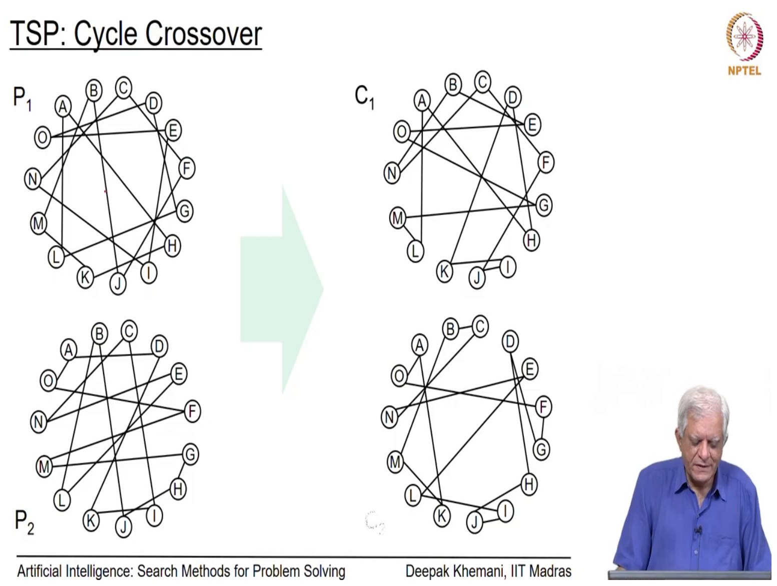
Now next time we will start with city L and so on till we have finished constructing all the cycles. At which point the two tours look like this as you can see they are colored according to cycles and we have five cycles in them.

The odd numbered ones are colored in the shades of blue-green and the even number ones are colored in the shake of shades of orange-red and the reason for that particular choice is because the way that we do cycle crossover is as follows that when we construct the first child, we take all the odd numbered cycles from parent 1 and then, we take the even numbered cycles from parent 2 and that gives us the first child essentially ok.

So, this process should be clear by now. We take the odd numbered cycles from parent 1 and even numbered cycles from parent 2 and you can see on the top the first child that has been constructed and it is a valid tour. There is every city exists exactly once and there is no repetition, and something has been carried forward from the two parent’s into this child. The second child is constructed in a similar fashion. We take the odd number cycles from parent 2 for child 2 and the even numbered from parent 1 essentially.

So, you can see for example, that this D, G has come from parent 1 which is a even numbered cycle. So, we have these two.

(Refer Slide Time: 07:14)

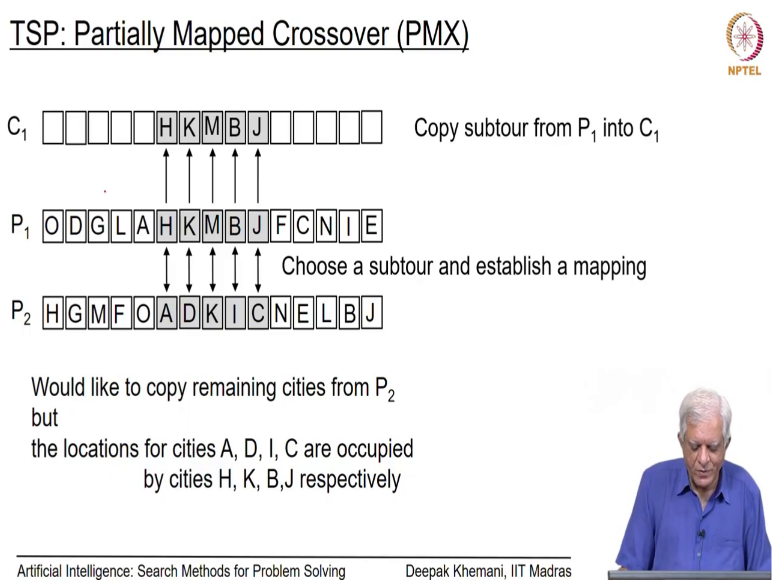

TSP: Cycle Crossover

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, if we look at the two tours that we just drew, and we have just drawn them randomly. We are not yet identified edge costs. Now you could imagine that if this is a real map, then the edge costs are in proportion to the actual Euclidean distance and this simply depicts that given this two parents P 1 and P 2, if we do cycle crossover we get the two children C 1 and C 2 on the right hand side.

Now, remember that if you are doing genetic algorithms, then you may have let us say started with a population of size 100 which means that 50 times you would do this crossover for two parents which are selected and out of those, the best ones would be carried forward and so forth and therefore, there is a lot of diversity that would be played around with and hopefully the best towards would get selected for reproduction or in the next cycle.

(Refer Slide Time: 08:23)



TSP: Partially Mapped Crossover (PMX)

C_1 □□□□□ H K M B J □□□□□ Copy subtour from P_1 into C_1

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

Choose a subtour and establish a mapping

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

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

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

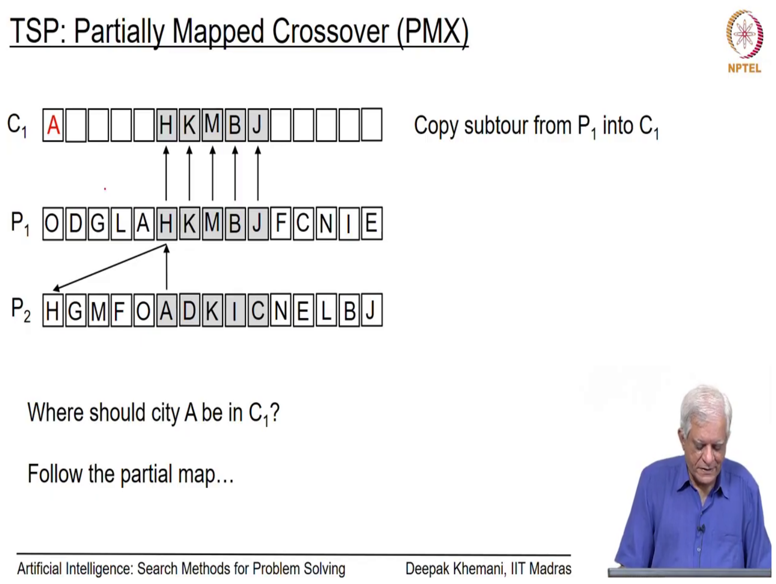
So, let us look at a couple of more crossover functions. One is called the partially mapped crossover and in this crossover, we first select the sub tour in both the tours which is in corresponding position.

So, that is shown here by the shaded cities H, K, M, B, J in parent 1 and correspondingly A, D, K, I, C in parent 2 essentially. So, having selected this sub tours, we have established a mapping between these cities and we would use this mapping to decide as to where to which cities to copy into each children.

So, the basic idea behind this is that, this sub tour that we have selected will pass on to one child each from each of the two parents and the remaining cities we will try as much as possible to get from the second parent essentially. So, that is the process we first copy the sub tour from parent 1 into child 1 and then we have to worry about where to copy the rest of the cities essentially.

So, we would like to copy the remaining cities from parent 2, but there is a little bit of a problem here if you see that since in parent 2, we have this cities A, D, K, I, C in those corresponding sub tour. Their positions are already occupied by this cities from parent 1. Of course, K is there in both so, it does not matter, but we have to worry about where we will place A, D, I and C in child 1. Because the position of A, D, I, C from parent 2 is already occupied by the sub tour that we selected from parent 1.

(Refer Slide Time: 10:19)



TSP: Partially Mapped Crossover (PMX)

NPTEL
C_1 A _ _ _ _ H K M B J _ _ _ _ _
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

Copy subtour from P_1 into C_1

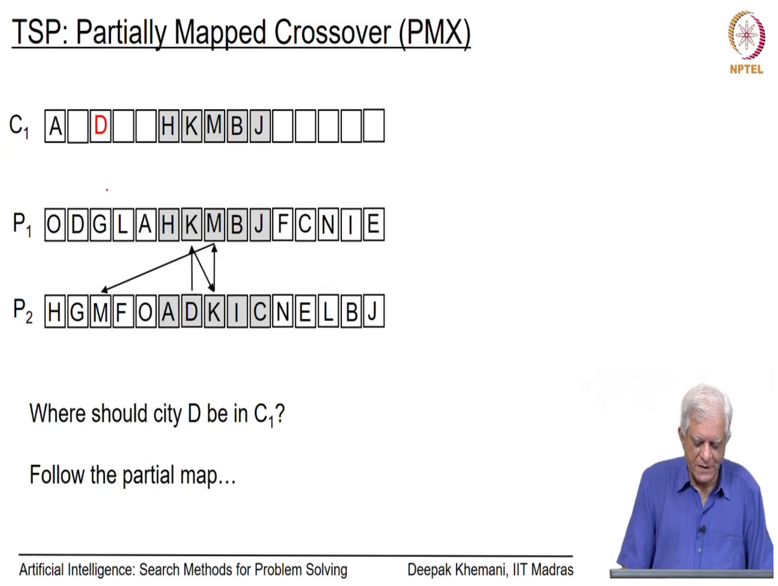
Where should city A be in C_1?

Follow the partial map…

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, let us go over them one by one. We start with city A and the question that we are asking is where should we put A in child 1. So, we use the partial map here and follow so, A is in the position of H. So, we see where H is and copy A in that position in child 1. So, you can see that we have kind of exploited the partial map to decide where to place city A.

(Refer Slide Time: 10:47)



TSP: Partially Mapped Crossover (PMX)

C1 A _ D _ _ H K M B J _ _ _ _ _
P1 O D G L A H K M B J F C N I E
P2 H G M F O A D K I C N E L B J
NPTEL

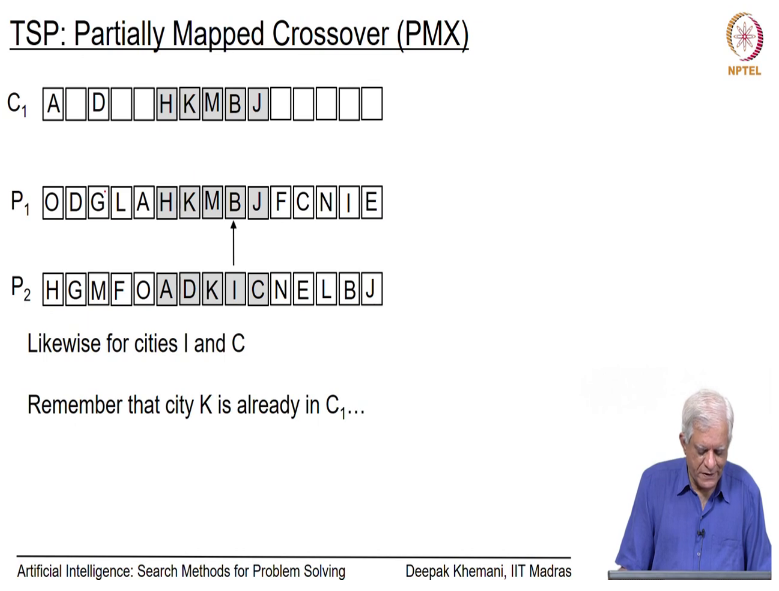
Where should city D be in C_1?

Follow the partial map…

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, let us look at the next city which is D. Where should we place D? In child 1, we do the same process. We follow the partial map from D we go to K, but K is already there in the sub tour. So, we follow from K to M and we find the location for M and then, that is where we place the city D essentially.

(Refer Slide Time: 11:12)



TSP: Partially Mapped Crossover (PMX)

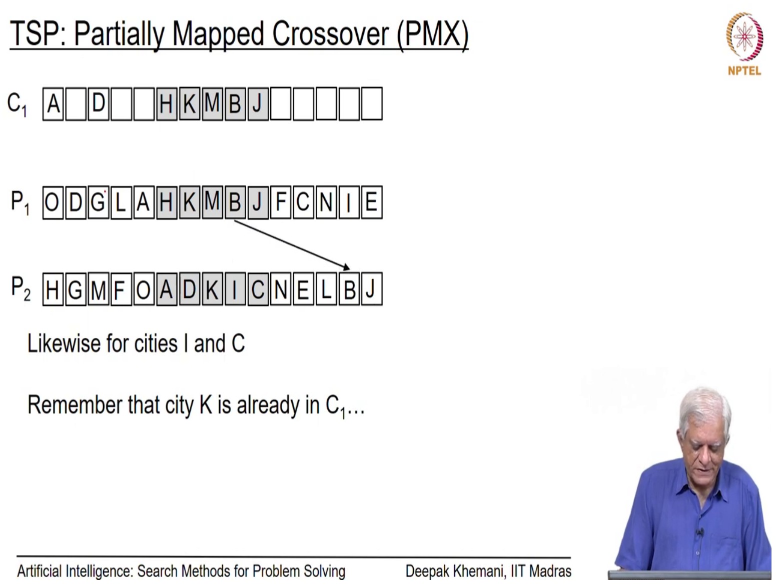
C_1	A		D			H	K	M	B	J					
P_1	O	D	G	L	A	H	K	M	B	J	F	C	N	I	E
P_2	H	G	M	F	O	A	D	K	I	C	N	E	L	B	J

Likewise for cities I and C

Remember that city K is already in C_1…

Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT Madras

(Refer Slide Time: 11:19)



TSP: Partially Mapped Crossover (PMX)

C_1 A □ D □ □ H K M B J □ □ □ □ □

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

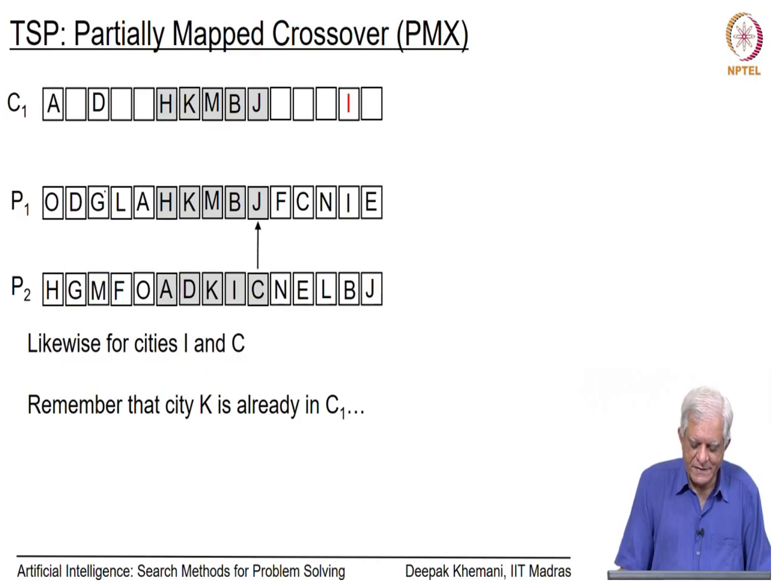
Likewise for cities I and C

Remember that city K is already in C_1…

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

In a similar fashion, we go and do the remaining two cities I and C. Remember that K is already there in child 1. So, we do not have to worry about it.

(Refer Slide Time: 11:21)



TSP: Partially Mapped Crossover (PMX)

C_1 A D H K M B J I

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

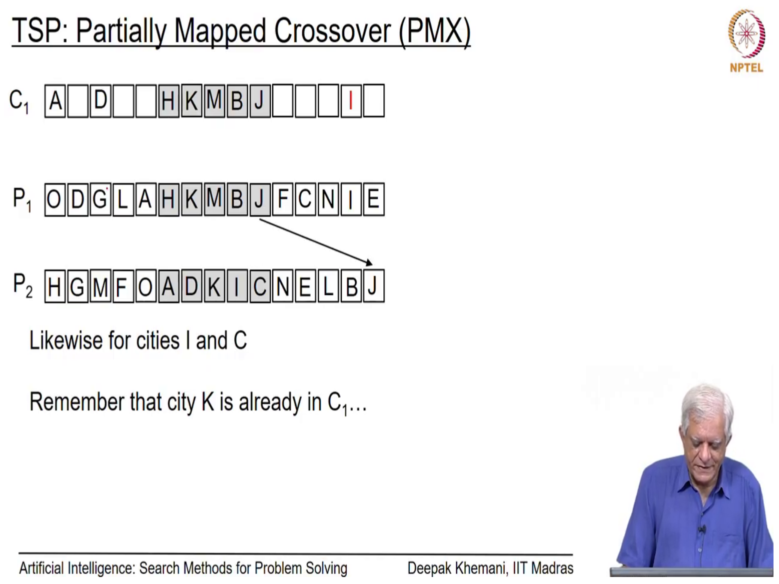
P_2 H G M F O A D K I C N E L B J

Likewise for cities I and C

Remember that city K is already in C_1...

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

(Refer Slide Time: 11:25)



TSP: Partially Mapped Crossover (PMX)

C_1	A		D			H	K	M	B	J				I	
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

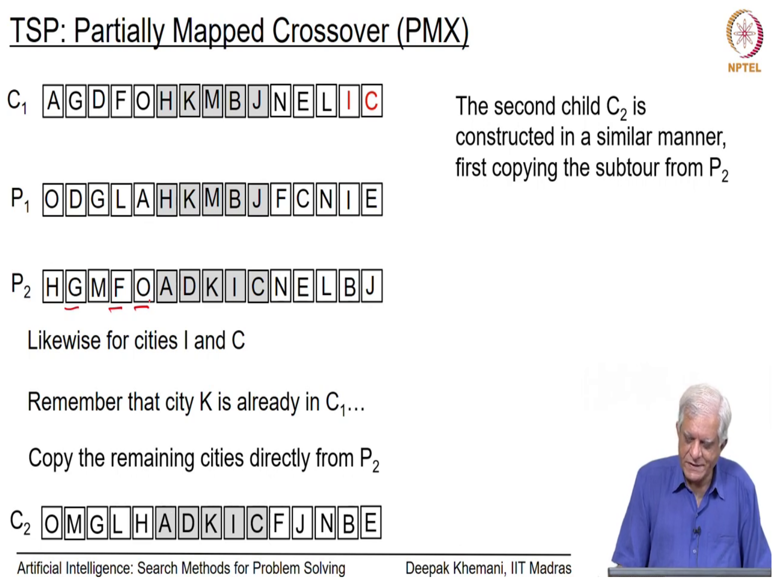
Likewise for cities I and C

Remember that city K is already in C_1…

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

It is such that I and C we need to. So, for city I we follow the trail and end up placing in the second last column and for city C, we place it end up placing in the last column. The remaining cities can now simply be copied from parent 2 into parent 1.

(Refer Slide Time: 11:36)



TSP: Partially Mapped Crossover (PMX)

C_1	A	G	D	F	O	H	K	M	B	J	N	E	L	I	C
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_2	O	M	G	L	H	A	D	K	I	C	F	J	N	B	E

The second child C_2 is constructed in a similar manner, first copying the subtour from P_2

Likewise for cities I and C

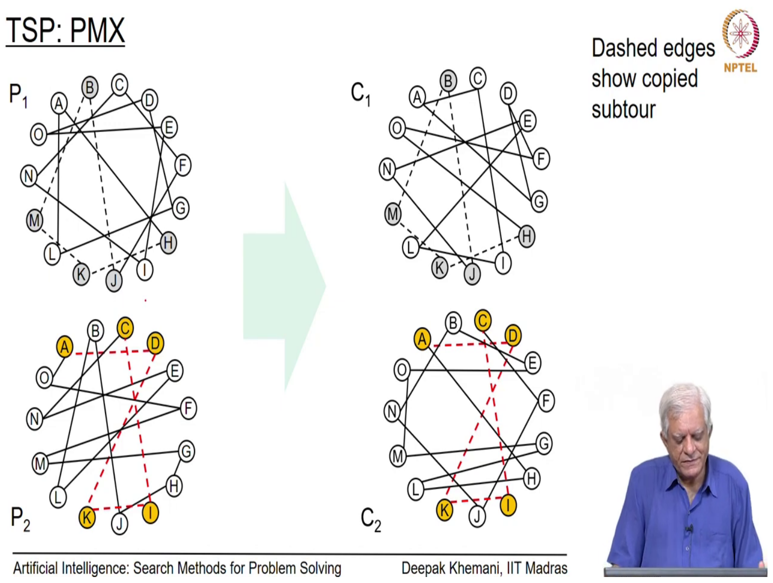
Remember that city K is already in C_1…

Copy the remaining cities directly from P_2

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, as we can see here, we are simply copying them from parent 2 to parent 1. So, these were the cities G and F and O and so on.

(Refer Slide Time: 12:05)



TSP: PMX

Dashed edges show copied subtour

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Madras

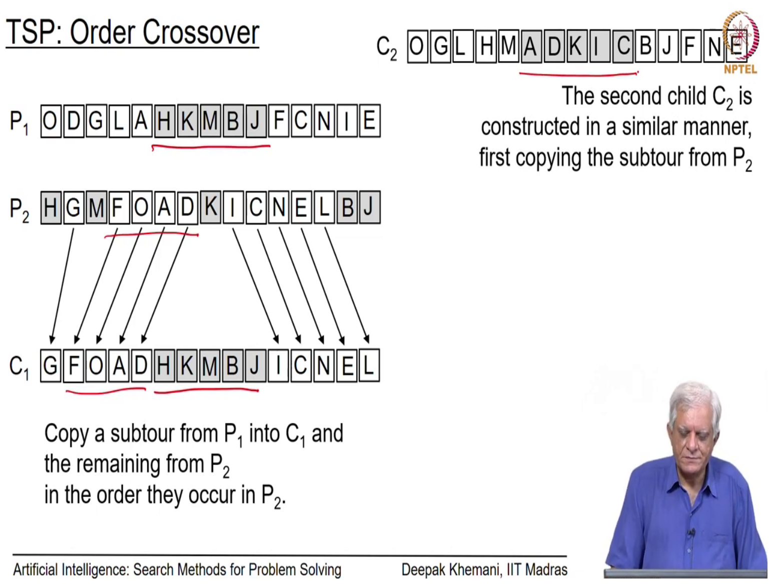
The second child is constructed in a similar fashion, that we first copy the sub tour from parent 2 and then, we follow the mappings to construct the tour essentially. This depicts the partially mapped crossover.

So, in the two tours on the left hand side, the shaded nodes are the sub tours that we are going to carry forward to each child and the dashed edges are the ones also which will get carried forward to the other child and the remaining part of the tours are constructed by this process that we just studied.

So, here you will see that child 1 inherits a sub tour from parent 1 and child 2 inherits a sub tour from parent 2 and the rest are constructed by this partially mapped process. So, now that we have this freedom to choose sub tours, you can imagine that we can introduce a heuristic component here and while making those or while making the random choice of the sub tour, we could try and select sub tours which are short essentially, which are composed of short edges.

And in that forward, in that manner we can try and pass them on to the children and the remaining is done by mixing of the things. So, intuitively it feels that you might get towards better tours by choosing the sub tours more intelligently.

(Refer Slide Time: 13:24)



TSP: Order Crossover

C_2 O G L H M A D K I C B J F N E

The second child C_2 is constructed in a similar manner, first copying the subtour from P_2

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

Copy a subtour from P_1 into C_1 and the remaining from P_2 in the order they occur in P_2.

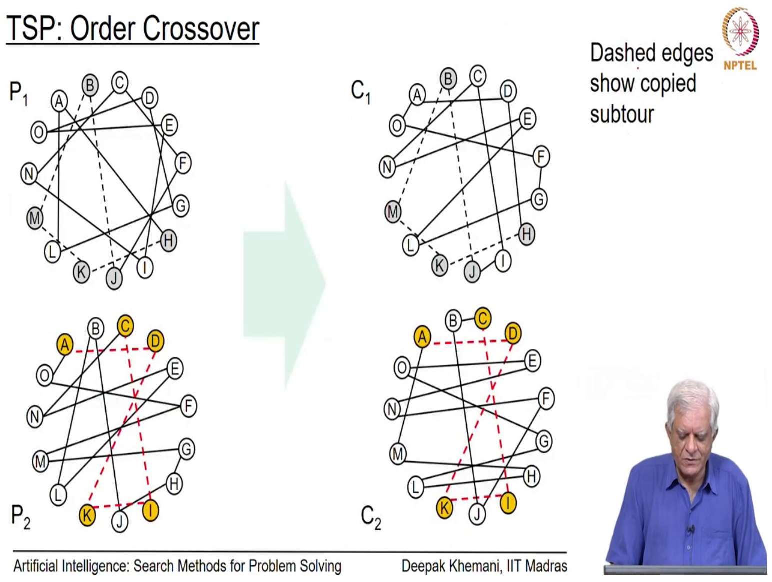
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

Another crossover which relies on sub tours is called the order crossover. So, this is a straightforward crossover that you copy the sub tour in this case the same H, K, M, B, J from parent 1 into child 1 as you can see this sub tour has been copied here.

And then, you look at parent 2 and ignore those cities H, M, K, B, J because they have already been copied and the remaining cities we copy them in the same order as they occur in parent 2 and this crossover is therefore, called order crossover because in the second that cities which are taken from the second parent they are taken in the same order. So, for example, F, O, A, D is here and F, O, A, D is also here essentially.

The second parent is constructed in a similar fashion and that is where you can see that this A, D, K, I, C was corresponding sub tours in parent 2 and that has been copied into this and the others have been copied in the order of children from all the cities from parent 1.

(Refer Slide Time: 14:39)


TSP: Order Crossover

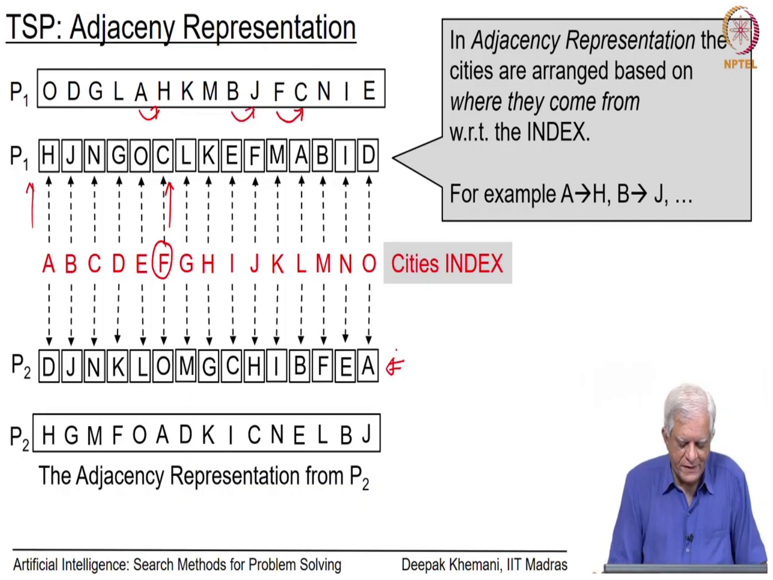
Dashed edges show copied subtour

Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT Madras

If you were to look at the tours that are constructed by the order crossover, then you can see that again like in P, M, X a sub tour has been copied from P 1 into C 1 and from P 2 into C 2 and the rest are done up by some process of mixing up.

In both these things, we have been extremely careful that we always produce children which are valid to us and that is where these crossover functions are have been carefully designed ok.

(Refer Slide Time: 15:11)



TSP: Adjaceny Representation

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

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

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

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

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

The Adjacency Representation from P_2

In *Adjacency Representation* the cities are arranged based on *where they come from* w.r.t. the INDEX.

For example A→H, B→ J, …

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, path representation is not the only representation that you can use for representing a tour. Another popular representation is called the adjacency representation and that basically works as follows: that you first construct a cities index.

So, you can see that there are these 15 cities in the traveling salesman problems and there we organize them in the index in this case in alphabetic order. So, A, B, C, D, E, F, G, H and so on and we use this index to access parts of the tour and in the adjacency representation, we create a representation which tells you from where you came from essentially.

So, if you look at this in the path representation, you going from A to H and in the adjacency representation, this has been shown by storing H in the first location and the first location signifies that from the first city in the index where did you go. So, since from A you went to H in the first location we store H.

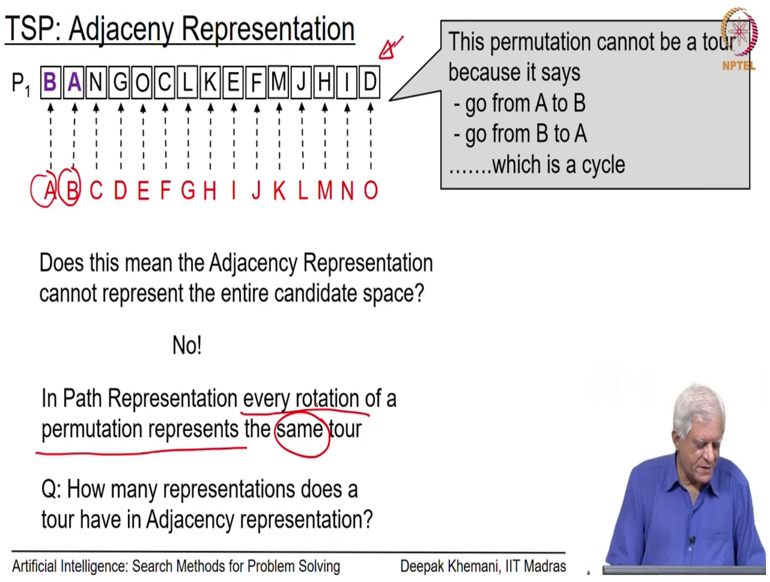
Likewise in the second location, we store J and the reason we do that is because you can see in the path representation that you have gone from B to J essentially and likewise for the rest of the cities, we have created a path representation, we have created a different representation which is the adjacency representation.

What is the advantage of this representation? The main advantage is that you can access because you know index representations are easier to access. You can access any segment of the tour quite easily instead of having to scan the entire tour.

So, for example, if you wanted to ask where in the given tour do you go from F, then in the past representation you would have to scan the entire tour till you reached F and then, you would figure out that from F you go to C. Whereas, in the index representation, you know that F where F is and then, you can simply go and figure out where you going to see. So, it becomes computationally easier to access specific edges in a tour.

We could create a adjacency representation for the second tour in our problem which was this H, G, M, F, O and you can see that we have got the adjacency representation here essentially.

(Refer Slide Time: 17:54)



TSP: Adjaceny Representation

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

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

This permutation cannot be a tour because it says
- go from A to B
- go from B to A

.......which is a cycle

Does this mean the Adjacency Representation cannot represent the entire candidate space?

No!

In Path Representation every rotation of a permutation represents the same tour

Q: How many representations does a tour have in Adjacency representation?

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

Now, there is one thing that one should be a bit careful of that look at this particular tour. If we create a permutation which looks like this B, A, N, G, O, C and so on. This permutation cannot be a tour because you are saying that you go from city A to city B and from city B you go to city A and that you can see is a cycle in the path representation.

Now obviously, you cannot have cycles in sub cycles in a tour because there is only one cycle and that should contain all the cities. So, one has to be a bit careful that not all permutations of cities. So, this is a permutation for example, but this is not a valid tour. So, not all permutations are valid tour.

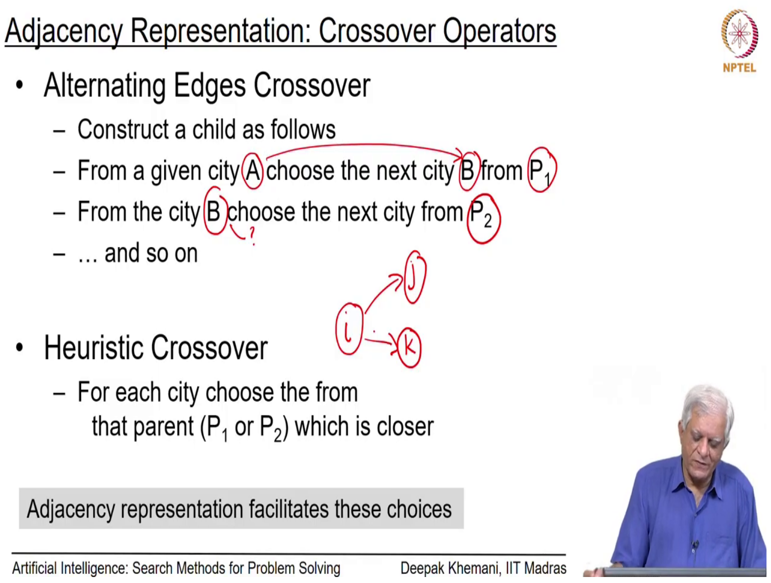
So, the question one might ask is does it mean that the adjacency representation cannot represent the entire candidate space because that is something that we would not like, we would like our search algorithm to access the entire space in search of the best candidate.

The answer to that fortunately is no, that it is not the case that this representation misses out on some tours. It is just the case that the path representation every rotation of a permutation represents the same tour essentially whereas, we cannot do rotations here because that would mean something else.

Because the first city in adjacency representation is the city you go to from the city A, the second city is the city you go to from city B. So, if you want to change, if you want to rotate that permutation then you would get a different tour. So, in that sense, the adjacency representation has a unique representation for every tour whereas, a path representation you can rotate the permutation and still get the same tour essentially.

So, I said unique, but perhaps you need to think a little bit about this and I will leave you with a question as to how many representations does the tour have in adjacency representation? Perhaps it is not exactly unique, but maybe you should just give a little bit of thought to it.

(Refer Slide Time: 20:19)



Adjacency Representation: Crossover Operators

- 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
  - … and so on
- Heuristic Crossover
  - For each city choose the from that parent (P_1 or P_2) which is closer

Adjacency representation facilitates these choices

Artificial Intelligence: Search Methods for Problem Solving — Deepak Khemani, IIT Madras

Now, how does this help adjacency representation? The crossover operators that we define over adjacency representations are as follows: that one is called the alternating edges crossover. So, what this says is that you construct a child as follows, you start from some given city some randomly chosen city and so let us say the city is A, then choose the next city B from parent 1.

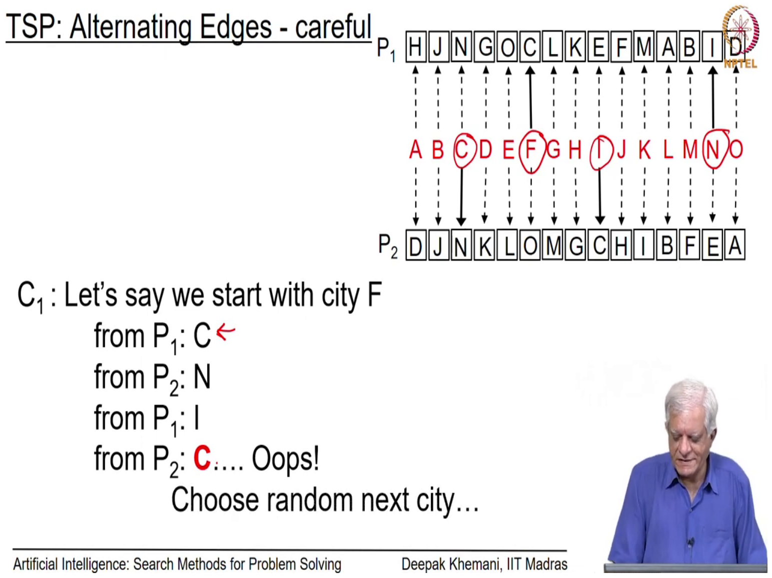
So, you are started with city A and then, you go to parent 1 and you find that from A you are going to B essentially in and this is can be easily found in adjacency representation as we have seen. Then from city B where should we go? We go to parent 2 to look for that essentially hence, so on and so forth.

Another crossover function which is kind of similar in nature is that when you want to go from a given city to another city, then choose the option from the two parents which is closer essentially.

So, if you are at city I and then parent 1 says that you go to city J and parent 2 says that you go to city K which one should you go to? Heuristic crossover says that choose the one which is shorter edge which is the shorter edge and go to the city which is closer essentially. In that sense, we are trying to drive the algorithm towards newer towards shorter tours essentially.

And as we have been talking, the adjacency representation facilitates these choices because you can go to the tour and immediately find out where you are going from a given city in this case you can say from I you are going to J in parent 1 and from I you are going to K in parent 2. This information can be accessed for past in adjacency representation.

(Refer Slide Time: 22:23)


TSP: Alternating Edges - careful

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

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

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

C_1 : Let's say we start with city F

from P_1: C

from P_2: N

from P_1: I

from P_2: **C**.... Oops!

Choose random next city…

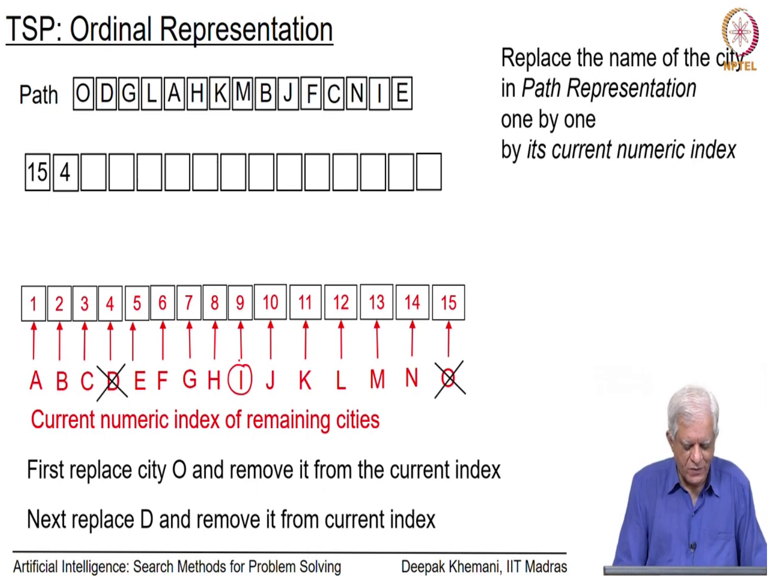
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

But there is a little bit of carefulness that you have to again exercise here and that is illustrated by this example. That supposing we do this alternating edges crossover and let us say that you start with city F which is here and as you can see from the arrow, you go from city F to city C having chosen it from parent 1.

Then when you go to when you have to decide where to go from city C you go to parent 2 and that takes you to city N and then from city N you go back to parent 1 to look for where to go and that gives you the city I and then, when you go from I you again look to parents 2 and there unfortunately you find that you have come back to city 1 to city C which is already there in the tour essentially.

So, city C is there, and you are getting it again. So, that; obviously, does not work and that is the problem both with alternating edges crossover and heuristic crossover. The solution to that is simply that at this point just choose a next city randomly and then carry on with it essentially.

(Refer Slide Time: 23:39)



TSP: Ordinal Representation

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

15 4

Replace the name of the city in *Path Representation* one by one by *its current numeric index*

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	~~D~~	E	F	G	H	I	J	K	L	M	N	~~O~~

Current numeric index of remaining cities

First replace city O and remove it from the current index

Next replace D and remove it from current index

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, let us look at one last representation and this is called as the ordinal representation. So, again let us consider our tour which we called as parent 1 in the path representation it says O, D, G, L, A; that means, you start with O go to D go to G go to L and go to A and then so on.

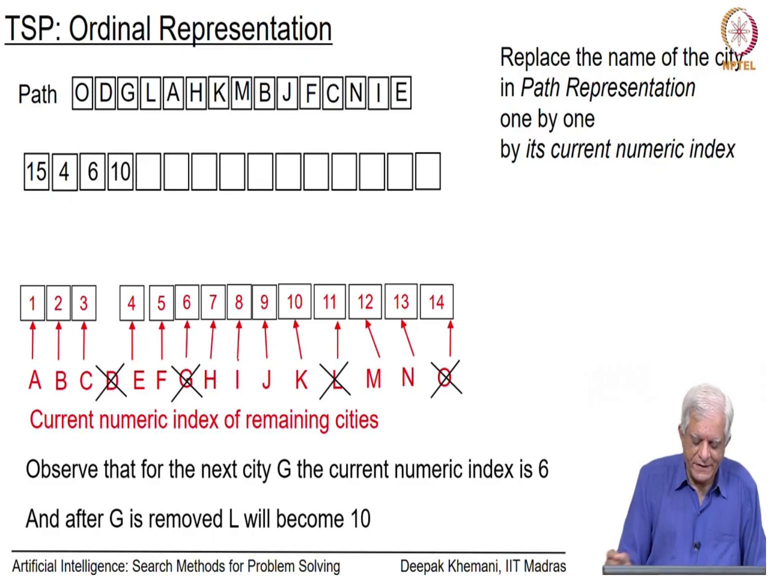
Like in the adjacency representation, we start with an index of the cities O, but what we do now is that we make a keep a numeric index. In some sense what this numeric index is saying for example, is that the city I is the 9th city in the index and city L is the 12th city and so on and so forth, city A is the 1st city. And we use this index numbers to create a new representation which is called as ordinal representation and we do it as follows: that we replace the name of the city in path representation and we do this one by one.

We start with city O, then we go to city D, then we go to city G and so on and we represent replace it by the current numeric index. So, we will see how what do you mean by this current numeric index.

So, let us start with the first city which is O and so, what we do is that we replace O with its value 15th because O is in the 15th position in the current index and then, we delete O from the index. In this case because O is the last city it does not matter too much, but in the next example when we go to city D, we find that city D is in the 4th place in the index. So, we will replace this symbol D with the value which is 4 because that is the place in the index and then, we will delete city D from the index.

Now, once we delete city D from the index, you will observe that city E has now become the 4th city and F has become the 5th city and so on. So, in that sense, the index is dynamic in nature and that is what we mean by the current numeric index essentially.

(Refer Slide Time: 26:10)



TSP: Ordinal Representation

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

15 4 6 10

Replace the name of the city in *Path Representation* one by one by *its current numeric index*

Current numeric index of remaining cities

Observe that for the next city G the current numeric index is 6

And after G is removed L will become 10

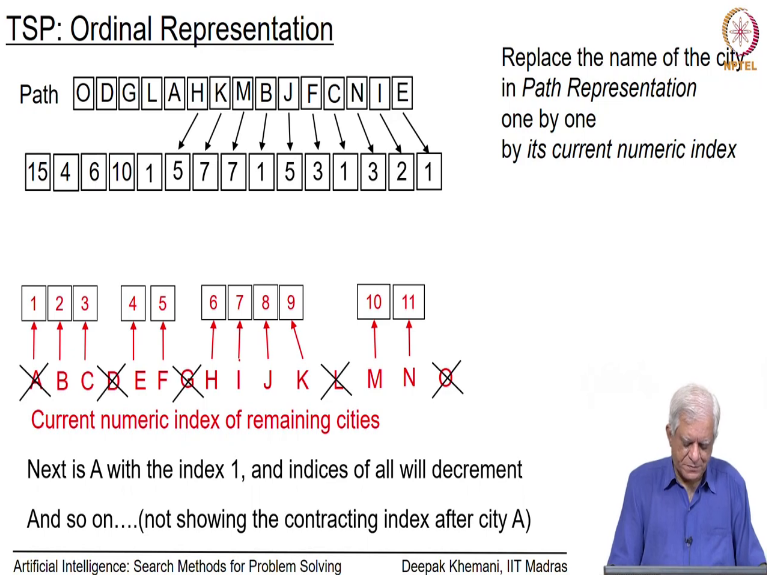
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, let us continue this process a little bit. We have already replaced O and D with 15 and 4 and this is what the index looks like. Notice that E has become 4, F has become 5, G has become 6 and so on and O which is anyway no longer there, but it would have been the 14th city if it was there instead of 15th.

So, the next city that we want to add is G and observe the G is in the 6th position in the index at this moment and therefore, when we do this, we copy the value 6 into where G was and you must remember that now G is removed from the index so, H would have moved to 6th position, I would be in the 7th position and J would be in the 8th position and so on. So, everybody would change by one. Everybody beyond G would change by one and that is what we use in the this thing.

So, after G is removed L will become 10. Initially L is 11 in this diagram, but after we have removed G, it would become 10 and therefore, we would copy 10 into the city.

(Refer Slide Time: 27:20)



TSP: Ordinal Representation

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

15 4 6 10 1 5 7 7 1 5 3 1 3 2 1

Replace the name of the city in *Path Representation* one by one by *its current numeric index*

Current numeric index of remaining cities

Next is A with the index 1, and indices of all will decrement

And so on….(not showing the contracting index after city A)

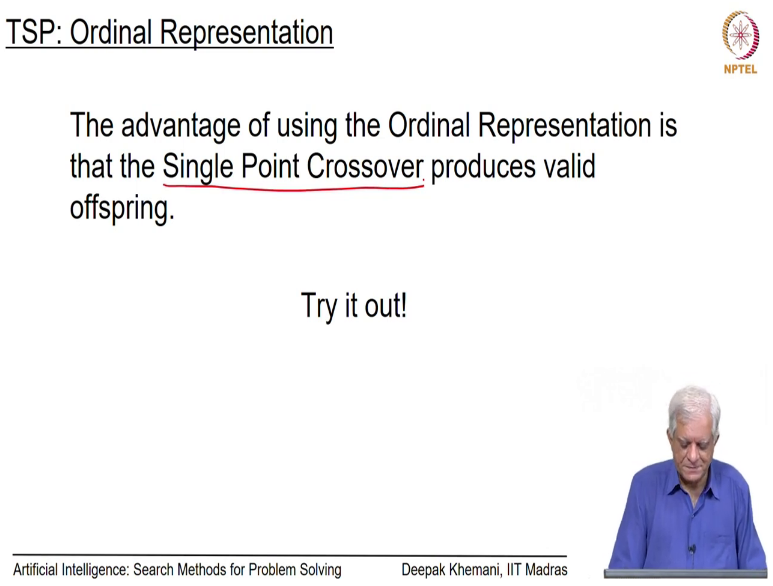
Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT Madras

So, we repeat this process, I will not go through the change in the index diagram, but you can imagine that this is what it would happen subsequently we have entered four cities here O, D, G and L and this is what the index looks like. We have 11 cities left in the index and we copy them one by one, but we will not make the changes in the diagram.

So, this is what happens A gets the value 1 then everybody else would get their index decremented. So, by the time that happens H has become 5 so, H would get the value 5 and so on essentially. So, as you can see one by one we copy the edges. We keep remembering remember that we have to keep changing the current numeric index.

As and when we keep adding a city to the ordinal representation, we must remove it from the index and therefore, look at the location for the next city in an appropriate place and as you can see the numbers have started decreasing now because more and more cities have vanished from the index and towards the end, it always comes into low single digit numbers essentially. So, this representation that we have just constructed is called the ordinal representation.

(Refer Slide Time: 28:48)



TSP: Ordinal Representation

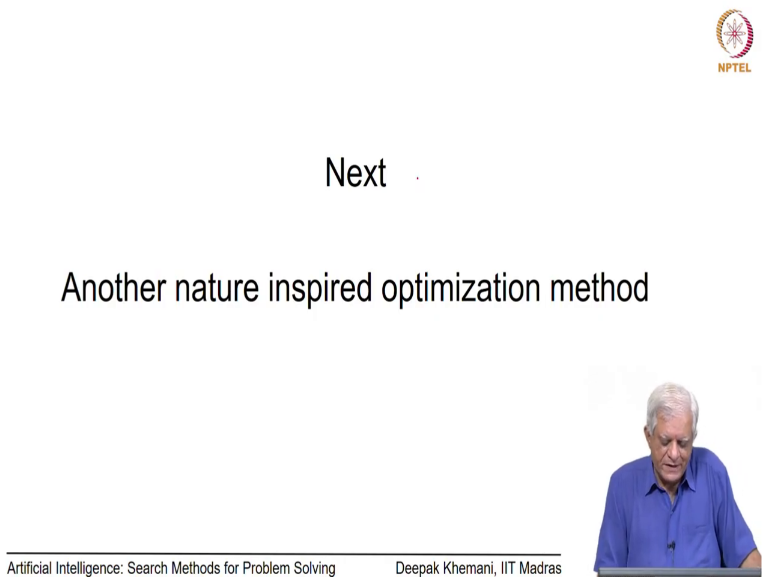
The advantage of using the Ordinal Representation is that the Single Point Crossover produces valid offspring.

Try it out!

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

As you can see that it is a little bit of a work to come to this representation. So, what do we gain out of this? So, what we gain out of this is that you can use the single point crossover over the ordinal representation. So, you take two tours convert them into ordinal representation and then, if it was single point crossovers what you get is valid offspring or valid tours essentially. So, you can think of that as an advantage and I would encourage you to try it out yourself with a small example ok.

(Refer Slide Time: 29:16)



Next

Another nature inspired optimization method

Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT Madras

So, we are done with TSP and genetic algorithms and we spent a little bit of time looking at genetic algorithms. In the next session, we will look at another population based optimization method which is also nature inspired essentially. So, we will look at the world of bees and ants and try to see if we can learn something from them in the process of optimization ok. So, we will see you in the next session.