

**Urban Transportation Systems Planning**  
**Prof. Bhargab Maitra**  
**Department of Civil Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 48**  
**Deterministic Traffic Assignment - II**

Welcome to Module F lecture 8. In this lecture we shall continue our discussion about Deterministic Traffic Assignment.

**(Refer Slide Time: 00:24)**

### Recap of Lecture F.7

---

- Deterministic Traffic Assignment
  - ✓ All-or-Nothing
    - Example
  - ✓ Incremental Assignment
    - Algorithm
    - Example

In lecture 7 we discussed about two assignment techniques one is All-or-Nothing assignment. And then we discussed also the merits and the demerits associated with this all-or-nothing assignment techniques, why the results might be wrong under which conditions or which scenarios? And then also indicated that where it could give a reasonably good result as well. Uncongested network or where the capacity constraint is not there.

And where the demand is very low as compared to the available capacity. And even after assignment also the shortest path remain shortest path. So, there under those scenarios it may give even good results. But majority case urban transportation network in Indian scenarios there is great imbalance between the demand and supply in the context of road transport. So, the congestions are there, the capacity constraints are there.

So, therefore the All-or-Nothing assignments are unlikely or more likely to produce unrealistic results. Now the effect or you know the large division what was happening can be to some extent handled to improve the results. If you apply Incremental assignment where instead of loading the whole network in one go, we are predetermined number we are deciding that in 4 or 5 or 6 increments or 10 increments we will do.

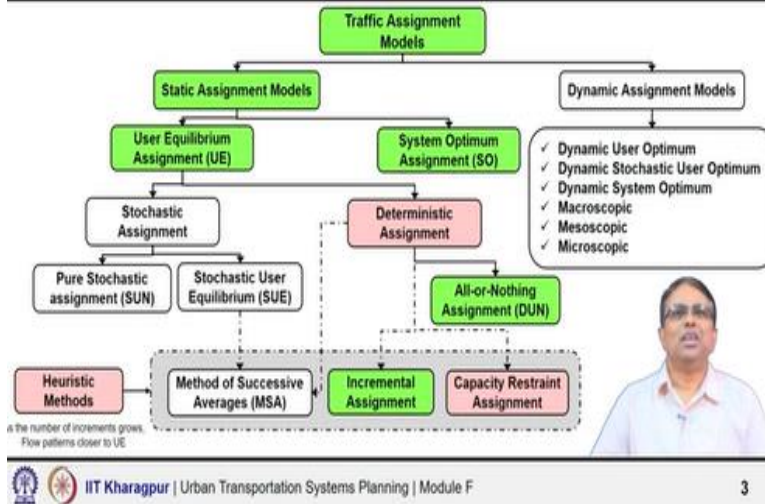
And then each increment a small fraction of the total load is assigned and every time we assign, we recalculate the shortest path. Then under that condition with that level of loading whatever is the shortest path. Next incremental load is assigned along the shortest path and then we update the flow or calculate the cumulative flow. Then again update the travel time calculate the shortest path apply the next incremental load then along the shortest path.

So, like this as we have seen that we can improve the solution to a great extent it you can improve the solution. But still the true sense user optimal equilibrium we are unlikely to get. Because in the last increment may distort the whole thing very drastically depending on at which point the road is operating and then even a small increment may have bigger impact if the loading is at or near the capacity.

But obviously incremental gives better result than all or nothing and incremental also use All-or-Nothing assignment only. But the travel times are updated after every increment and new shortest paths are and shortest paths are recalculated. The path may be new but may be old also the earlier one also may come out eventually to be the shortest path. But we recalculate the shortest part.

**(Refer Slide Time: 03:57)**

## Approaches to Traffic Assignment



Now we go to another kind of assignment where the capacity constraint assignment because as we have been talking about the capacity constraints in urban road network. And we said that in on many occasions some links may have you know or some paths may have less capacity than the actual demand. So, that is another reason why the demand will be distributed to multiple roads, because of course the congestion effect will make the travel time to go higher. But even if suppose the travel time you know even after that the still it is travel time wise cheaper.

If I just assume it like that, then also so much demand cannot simply be accommodated by the path. So, that may be one reason why you know the demands are going to be distributed to multiple paths. Now this capacity constraint assignment directly or indirectly considers this aspect considered the aspect of capacity may be through a proper link cost function or demand versus cost or time function.

But the capacity is considered directly or indirectly. So, that is why we say the constraints in the capacity which actually and the aggravated congestion that effect is duly considered. Those techniques we will call capacity resistant assignment.

**(Refer Slide Time: 05:41)**

## Deterministic Traffic Assignment

### Capacity Restraint Assignment

- Attempts to approximate an equilibrium solution by iterating between all-or-nothing traffic loadings and recalculating link travel times based on a congestion function that reflects link capacity
- Unfortunately, some times, this method does not converge with flow flip-flops between some links but no loading on other links
- A remedy to this situation is the use of modified capacity restraint algorithm



So, let us look at the capacitor resistance assignment. It attempts to approximate an equilibrium solution by iterating between all-or-nothing traffic loading and then recalculating link travel time based on a congestion function that reflects the ring capacity. Please observe this part. Based on a congestion function travel time function cost function whatever you say that reflects the capacity. So, capacity is somehow considered inside that calculation.

So, I calculate recalculate the travel time, so my capacity is really considered there. Now unfortunately the when the capacity resistant assignment was developed initially it was realized that sometimes not always. This method does not converge, why? Because it results to a situation where the flow flip flops may occur. So, in one iteration a is getting loaded another iteration b is getting loaded, then again, a then again b then again a again b.

So, whatever number of iterations you keep on doing it will keep flip flopping from one path to another path only two paths are maybe you know handling the flow. And remaining paths are no more handling the no more accommodating any flow the no flow is coming to other parts. So, that sometimes is unrealistic. And every problem we faced its research so people try to find out a solution also. So, the modified capacity resistant algorithm also was developed.

This is the problem the flow flip flop is happening. So, how to overcome? Researchers come out with alternative ideas. So, one such idea is what we will discuss in this case is modified capacity

resistant algorithm.

(Refer Slide Time: 07:45)

## Deterministic Traffic Assignment

• The algorithm can be summarized as follows

**Step 0: Initialization**

✓ Perform all-or-nothing assignment based on  $t_a^0 = t_a(0)$ ,  $\forall a$

✓ Obtain a set of link flows  $\{x_a^0\}$ . Set iteration counter  $n = 1$

**Step 1: Update:** Set  $t_a^n = t_a(x_a^{n-1})$ ,  $\forall a$

**Step 2: Network loading:** Assign all trips to the network using all-or-nothing based on travel time  $\{t_a^n\}$ . This yields a set of link flows  $\{x_a^n\}$



So, let us try to see quickly that how the algorithm works. So, first part is the initialization simply perform all-or-nothing assignment based on the zero flow. Because initially we have not loaded the network. So, we know the zero-flow travel time and then we say you know accordingly assigned everything along the shortest path all demand. And that set the counter that is the number of iterations as 1.

Then update the time in step one with the new flow you know so the travel time is a function of flow. So, we update the travel time then assign all trips again following the all-or-nothing assignment based on this travel time and according the shortest path. Then the previous if it is different from the previous link or previous part then previous paths is unloaded kind of thing. And the whole loading is now done on this path.

(Refer Slide Time: 08:47)

## Deterministic Traffic Assignment

---

**Step 3: Convergence:** If  $\max_a \{|x_a^n - x_a^{n-1}|\} \leq k$ , stop (the current set of link flows is the solution). Otherwise, set  $n = n+1$  and go to step 1;

'k' is a predetermined tolerance selected especially for each problem based on the desired degree of accuracy.

This convergence test is based on the maximum change in link flow between successive iterations. Other criteria can also be used.



So, like that we can continue and there are so many different ways of converging somebody may say that with fixed number of iterations we will do, that also one can do. Actually, you know this these are not just one algorithm there are so many variations of each of these algorithms, may be fundamentally everybody is considering the capacity. But then you know little bit it may vary even if you see different books for different algorithms.

Specially, the modified one all-or-nothing you know will remain same because it is so simple that there cannot be any variation. But how to consider capacity how to bring the link condition effect? There are so many other things also could be there everything we are not discussing in this course. And maybe there could be things also which I also do not know maybe a slightly different form is available somewhere.

But the fundamentals all the basic approaches that we are covering in this course. So, it will not be very different, but exact mathematical function the link cost function exact way of updating it may vary from you know one approach to another approach minor variations are always possible. So, you can have a fixed number or sometimes people say an error I mean if you consider a multiple origin multiple destination.

Then even if you are doing that not always for every path a new thing will come up may not come up. And you therefore keep a threshold that when from one iteration to another iteration

the maximum change is within this permissible limit. So, that means you know not much change is happening and whatever change is happening that is acceptable for you, because you consider that change as a negligible change.

So, that way also one can decide that where to stop. So, basically the idea is that you know assign the load again and again. You know based on the you know the calculated newly calculated shortest path the complete loading you do. But as I said sometimes, I want to show the problem here not the solution, because solution will show you with an example of the modified capacity restraint.

(Refer Slide Time: 11:17)

**Deterministic Traffic Assignment**

---

**Example**

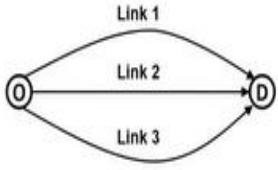
Obtain the flow patterns over the network shown in Figure. The volume-delay curves are as given below.


$t_1 = 17 [1+0.3(x_1/4)^2]$  time units

$t_2 = 16 [1+0.5(x_2/5)^3]$  time units


$t_3 = 12 [1+0.6(x_3/7)^3]$  time units

$x_1 + x_2 + x_3 = 12$  flow units





---

 IIT Kharagpur | Urban Transportation Systems Planning | Module F
7

So, let me try to solve so an example where such kind of this flip flop situation may occur. So, let us take these three parts Links 1, Link 2, Link 3. Remember that you may not get the exact feel of all these assignments. Because assignments are done on a network basis. So, you have multiple origin multiple destination between every od pair some demand is happening and there may be so many links large number of links in the network.

Some links even may be common to some paths. So, there again how to consider lot of things are there. But just to explain you to give a better feel than just showing the steps you know  $t$  a or  $x$  a update  $x$  a  $t$  as a function of  $x$  a. And then go there next step. So, we wanted to take some simple example where we can explain you really how it is working. But you have to understand that the

real network does not work only with one origin and one destination and just you know three link alternative link.

These are simplistic representation. So, that additional you know some kind of challenges and some kind of considerations you have to keep in mind when you apply to a realistic network. Nevertheless, let us take this three, this small example with the three links and the travel time flow functions are there.

(Refer Slide Time: 12:54)

## Deterministic Traffic Assignment

### Solution

Iteration Number	Algorithmic step	Link 1	Link 2	Link 3
0	Initialization	$t_1^0=17$ $x_1^0=0$	$t_2^0=16$ $x_2^0=0$	$t_3^0=12$ $x_3^0=12$
1	Update Loading	$t_1^1=17$ $x_1^1=$	$t_2^1=16$ $x_2^1=12$	$t_3^1=48$ $x_3^1=0$
2	Update Loading	$t_1^2=17$ $x_1^2=0$	$t_2^2=126$ $x_2^2=0$	$t_3^2=12$ $x_3^2=12$
3	Update Loading	$t_1^3=17$ $x_1^3=0$	$t_2^3=16$ $x_2^3=12$	$t_3^3=48$ $x_3^3=0$
.....		....	.....	....

The algorithm **does not converge**, as the flow **flip-flops** between links 2 and 3, whereas link 1 does not get loaded at all



And what we try to do the initialization we know that link 3 here is the cheapest with travel time as 12. So, all 12-unit flows which we wanted to assign we assign it through that. Then the travel time on link 3 becomes 48 with the revised load. So, now the shortest path becomes link 2. So, again this 12 units we load into link 2 and link 3 not get loaded. So, link 3 does not get loaded, so link 3 becomes come back to 12 only. But link 2 now become 126.

So, now you next iteration you load again the whole demand into link 3. And you can sync here and it is get link 3, then link 2 then link 3 again link 2. And whatever may be your number of iterations it will keep going from link 2 to link 3, link 3 to link 2. So, that is the kind of flip flop of flows that what we mentioned. So, you know this kind of problems may sometimes occur not always.



Because when you take a realistic network multiple origin destination it may converge. So, sometimes it gives a problem. So, to overcome this the modified algorithm came.

(Refer Slide Time: 14:14)

## Deterministic Traffic Assignment

### Modified Capacity Restrained Algorithm

**Step 0: Initialization:** Set the number of iterations (N) (including iteration 0) after which algorithm is to be terminated;  $N \leq 4$

Perform all-or-nothing assignment based on  $t_a^0 = t_a(0)$ ,  $\forall a$ . Obtain  $\{x_a^0\}$ .  
Set  $n = 1$

**Step 1: Update:** Set  $T_a^n = t_a(x_a^{n-1})$ ,  $\forall a$

**Step 2: Smoothing:** Set  $t_a^n = 0.75t_a^{n-1} + 0.25T_a^n$ ,  $\forall a$

**Step 3: Network loading:** Perform all-or-nothing assignment based on travel time  $\{t_a^n\}$ . Yields  $\{x_a^n\}$



Now what the modified algorithm does? Here also the initialization is set the number of iterations in the beginning that how many iterations you want to perform. And normally it is not taken less than four iterations many cases. It is more even it could be even higher much higher. So, perform all-or-nothing assignment initially based on the zero flow. So, you know the zero-flow travel time or alternative path so just assign along the zero flow.

Now the step one you update the flow. So, what we are doing we are calculating we know the flow. So, accordingly we update the travel time. Then, here onwards the things are deviating little bit, what we thought? We thought that the new flow. So, the travel time exactly whatever we are getting as per the new flow we are taking and going to the next step loading we are doing based on this travel time here it deviates.

We are doing some kind of smoothing. What we are doing? Just see the interesting in the nth iteration whatever may be the based on this loading in nth iteration whatever is the travel time now with the loading. We are taking 25% weightage of that and 75% of the weightage. 75% weightage on travel time which were there in the earlier equation. So, that means you are not drastically changing the time.

So, some small change of travel time becomes smaller as compared to the previous one, does not change the time all on a set  $n$ . Then with this smoothing whatever is your travel time. Now I am finding shortest path and doing the all-or-nothing assign based on this travel time value. That is the only difference.

**(Refer Slide Time: 16:29)**

### Deterministic Traffic Assignment

- Step 4: **Stopping rule:** If  $n+1 = N$ , go to step 5; otherwise, set  $n = n+1$  and go to step 1
- Step 5: **Averaging:** Set  $x_a^* = \frac{1}{N} \sum_{l=0}^n x_a^l \forall a$  and stop



And then when you find that your predetermined  $n$ th iteration is done. Then you stop and then you calculate what is the average flow or average demand assigned to different routes. So, sometimes you might have assigned to one part the demand, next you might have assigned to another path next, we might have you have assigned to another part. Maybe some paths multiple times you have assigned.

So, now take in all four iterations average how much loading I have done in every iteration for every path. That is the load which is taken.

**(Refer Slide Time: 17:13)**

## Deterministic Traffic Assignment

### Solution: Using Modified Algorithm (CRA)

Iteration	Algorithmic step	Link 1	Link 2	Link 3
0	Initialization	$t_1^0=17$ $x_1^0=0$	$t_2^0=16$ $x_2^0=0$	$t_3^0=12$ $x_3^0=12$
1	Update Smoothing Loading	$T_1^1=17$ $t_1^1=17$ $x_1^1=0$	$T_2^1=16$ $t_2^1=16$ $x_2^1=12$	$T_3^1=48.3$ $t_3^1=21$ $x_3^1=0$
2	Update Smoothing Loading	$T_1^2=17$ $t_1^2=17$ $x_1^2=12$	$T_2^2=127$ $t_2^2=44$ $x_2^2=0$	$T_3^2=12$ $t_3^2=18$ $x_3^2=0$
3	Update Smoothing Loading	$T_1^3=63$ $t_1^3=25$ $x_1^3=0$	$T_2^3=16$ $t_2^3=37$ $x_2^3=0$	$T_3^3=12$ $t_3^3=17$ $x_3^3=12$



Let me try to show you that example the same problem. Now initially the travel time is 12  $t_0$  travel time that is the initialization. So, all 12-unit flows are assigned to link 3. Now link 3 travel time becomes based on this 12 unit flow the link 3 travel time becomes 48.3. And then what we did in the capacity restraint? 16 is the new thing, so everything assigned on 16. So, go back you see that it was 48 we have not considered the decimal here.

And that was the travel time we consider 17, 16, 48. And comparing these three, the shortest was link 2, so we assigned it to link 2. Here the difference is we are applying the smoothing. So, what we are doing? 0.75 of previous time that means 0.75 on  $t_0$  plus 0.25 on this update in the present iteration. So, the first case the first you know initialization that is 17 into 0.75 plus 0.25 into the second-row iteration 1 that update value first row in that that 17.

Obviously both 17 so whatever weightage is you give it even 17. So, the change will obviously happen to link 3. We are not going ahead with 48.3 or if I round it off it, we may consider 48, but what we are doing 0.75 into 12 + 0.25 into 48. That value we are taking, please see that 0.75  $T_{n-1}$  + 0.25 into whatever update you have got now that travel time value we have taken. So, exactly you do like that, then whatever is the travel time.

Now with the smoothing based on that in this case also in iteration one. It is coming out to be link 2, so we assign it to link 2, then again link 2 with all 12. It becomes 127 but we will not do

the assignment based on 127. What we will do? 16 the previous iteration time into 0.75 plus 0.25 into 127. Similarly, this for link 3 now the load is 0 in the previous iteration to the time is with 0 load it should become 12.

But we do not make it 12. 0.75 of the previous one which was  $21 + 0.25$  into 12. So, it becomes 18. Now you see now I compare 17, 44, 18 and then the next loading I will do. So, the next 12 gets loaded in link 1 like that we do all the four steps.

**(Refer Slide Time: 20:46)**


### Deterministic Traffic Assignment


---

Average of above four steps

	Link 1	Link 2	Link 3
Average	$x_1^* = 3.0$ $t_1^* = 19.9$	$x_2^* = 3.0$ $t_2^* = 17.7$	$x_3^* = 6.0$ $t_3^* = 16.5$

- For large networks, number of minimum-path computations is the main computational burden, and thus traffic assignment algorithms should be compared in terms of efficiency for a given number of minimum-path computations



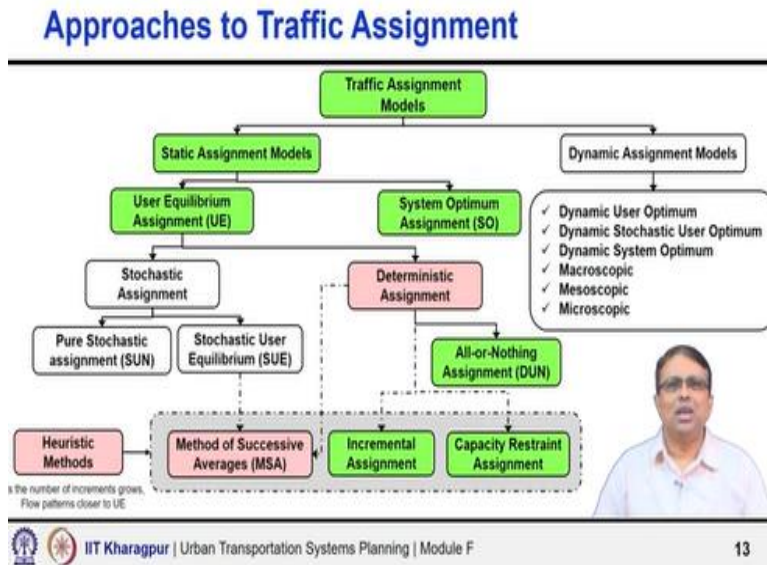

IIT Kharagpur | Urban Transportation Systems Planning | Module F
12

And then calculate the average flow and average time. Average flow first and then with that average flow what is the actual travel time you can see it is not exactly equal but definitely much better solution. So, for large network the number of minimum path computation is the main computational burden and thus traffic assignment algorithm should be compared in terms of efficiency for a given number of minimum path computation.

So, that is one you know aspect what you should bear in mind. But I hope you have understood these steps. So, predetermined step updating the travel time we are releasing so that means it does not go back to the original one. There is a dependency of what it is now based on the current load 25% weightage and 75% weightage on what was the updated travel time after smoothing. That we take and accordingly we calculate the travel time after smoothing in the present iteration.

And these travel times are used for calculating the shortest path and doing the all-or-nothing assignment in this step or in this iteration. That is the basic difference. And then finally we calculate the average flow assigned or average demand assigned to each path.

**(Refer Slide Time: 22:22)**



So, I am happy that we are getting more and more boxes as green color. So, that means the we are progressing and you know those boxes already we have discussed. So, now the next is method of successive average. This is also one of the heuristic methods like incremental assignment capacity restrained assignment. It is heuristic. So, we are trying to reach to an approximate solution. So, let us discuss about the method of successive average and then that will make the deterministic assignment part complete the whole discussion will be over.

**(Refer Slide Time: 23:04)**

## Deterministic Traffic Assignment

### Method of Successive Averages

- MSA were developed, at least partially, to overcome the problem of allocating too much traffic to low capacity link
- It can be describe following steps:

Step 0: **Initialization**. Perform a all-or-nothing network loading based on a set of initial travel times  $\{t_a^0\}$ . This generates a set of link flows  $\{x_a^1\}$ . Set  $n = 1$

Step 1: **Update**

$$\text{Set } \{t_a^n\} = t_a(x_a^n), \forall a$$



So, method of successive average were developed at least partially to overcome the problem of allocating too much traffic to low capacity link and doing some kind of literally balancing. You assign more has gone so take something put it here. Or now this has become more so take something put it there, still it is let to take little bit more and put it there like that you know trying to balance, see when the travel time become same.

So, it can be explained through the following steps. The first step is the initialization perform all-or-nothing network loading based on a set of initial travel time. That is, you know  $T_a^0$ . This generate a set of link flows, now what we are doing? We are updating the travel time.

(Refer Slide Time: 24:04)

## Deterministic Traffic Assignment

Step 2: **Direction finding**. Perform a all-or-nothing network loading procedure based on the current set of link travel times,  $\{t_a^n\}$ . This yields an auxiliary link flow pattern  $\{y_a^n\}$

Step 3: **Move**. Find the new flow pattern by setting

$$x_a^{n+1} = x_a^n + \frac{1}{n}(y_a^n - x_a^n) \text{ with } 0 \leq 1/n \leq 1$$

Step 4: **Convergence criterion**. Calculate a new set of current link costs based on  $x_a^{n+1}$ ; If the current link costs are converged, stop;

Otherwise set  $n = n + 1$  and go to step 1



And then doing also the direction finding I shall explain this first in this form. And then take an example and then you will be it will be much more clear to you when I take the example. So, perform all-or-nothing network loading procedure based on the current set of travel time. And these yields to a flow will assign it to the shortest path. You remember incremental assignment the modified capacity restrained assignment we did not change the travel time drastically.

So, we set travel time and then we did some kind of smoothing. So, here also whatever you know first we assigned if it is shortest assigned 2000 flow. But that is actually called auxiliary flow. So, then based on that auxiliary flow and whatever was my actual flow how I now try to update the flow. Somewhat conceptually you know there we do it with the time part, and here it is the flow that is getting adjusted.

So, we are not saying the auxiliary flow means new, next the complete loading is along the shortest path. But we are not saying that means my actual link flow is the whole flow which has come now. It has got some kind of interdependency based on whatever was the flow in the as per my last iteration or the most recent iteration. And whatever is the auxiliary flow now accordingly we generate it.

And then again keep on doing these things and then finally as long as we are till the time, we are happy in terms of getting the equal travel time and with reasonable flow distribution. So, let us take it an example.

**(Refer Slide Time: 26:08)**

## Deterministic Traffic Assignment

---

### Example

- Consider a case of a bypass and a single town-center route with a total demand of 2000 trips,  $C_b = 10 + 0.0065V_b$  and  $C_t = 8 + 0.01V_t$



Consider here for simplicity again we have taken only two routes two alternative routes. So, consider a kit of bypass and a single town center route with a total demand of 2000 trips and we want to distribute it using this two-cost function. You can clearly see here the travel time is zero flow travel time is research along the town route which is normally the shorter. But the sensitivity is much higher.

You can see as the volume increases the travel time increases every unit increase in volume by 0.01. Whereas the bypass route is a longer one but may be high capacity route. So, every unit increase in the demand or flow  $V$  the change in the cost is only 0.0065. So, that is a typical thing. The bypass route is a high capacity route longer but high capacity road does not get congested so fast. But the town route is the shorter one but often faster get congested.

**(Refer Slide Time: 27:18)**



## Deterministic Traffic Assignment

Solution:

Iteration (n)	Step Size	Travel times		Auxiliary Flow $y_a^n$		Link Flows $x_a^{n+1}$	
		Town	Bypass	Town	Bypass	Town	Bypass
Initialize		8.00	10.00	-	-	2000	0
1	1.00	28.00	10.00	0	2000	0	2000
2	0.50	8.00	23.00	2000	0	1000	1000
3	0.33	18.00	16.50	0	2000	667	1333
4	0.25	14.67	18.67	2000	0	1000	1000
5	0.20	18.00	16.50	0	2000	800	1200
6	0.17	16.00	17.80	2000	0	1000	1000
7	0.14	18.00	16.50	0	2000	857	1143
8	0.13	16.57	17.43	2000	0	1000	1000
9	0.11	18.00	16.50	0	2000	889	1111
10	0.10	16.89	17.22	2000	0	1000	1000
11	0.09	18.00	16.50	0	2000	909	1091
Travel time at convergence		17.09	17.09				

$$x_a^{n+1} = x_a^n + \frac{1}{n}(y_a^n - x_a^n)$$

$$x_{by}^{n+1} = 1000 + \frac{1}{3}(2000 - 1000) = 1333$$



So, that is what is the distribution. Look at this first we know that free flow travel time is 8 and 10 on town and bypass. So, first no auxiliary flow directly the link flows are 2000 and 0. Then with 2000 and 0 we calculate the travel time which are now town route is 28 and the bypass does not get loaded in the initial iteration, so it remains 10. So, now 8 and 28 and 10, so the bypass is cheaper.

So, my auxiliary flow is assigned to bypass you can see 0 is assigned to town in iteration 1 and 2000 is assigned to bypass so, that is the auxiliary flow. But then what is my update of flow? Now my next update of flow, it is actually you can see the link flows are for  $x_a^{n+1}$ . That means actually for the  $n+1$  iteration this is the update. But  $y_a$  is for the  $n$ , so based on the auxiliary flow we are updating now actually the link flow.

But I have presented it in the iteration number wise 1, 2, 3, 4 like that 11 iterations are presented. So, the link flows are shown as  $x_a^{n+1}$ . So, you know that it is first the auxiliary flow and then based on that you do the update and then going to the next iteration what is the travel time again. Look at this update. That is what is interesting what we are saying our bypass is 2000 so I do not make you know this 0 and 2000 not just solely based on auxiliary flow.

Incidentally it comes 0 to 2000 only. But it is based on that equation. Though, what we are saying? Look at this town flow  $x_a^n$ , it was 2000. So,  $n+1$  we are calculating  $x_a^n$  was  $2000 + 1$

by  $n$ ,  $n$  here is 1. What is  $y_n$ ?  $y_n$  auxiliary flow is  $0 - x_{a,n}$ ,  $x_{a,n}$  again is 2000. So,  $2000 - 2000$ , it becomes 0. So, 0 and 2000, so with this 0 and 2000 flow we now again calculate the travel time.

We find that the town is 8, because it is 0 flow and the bypass with 2000 flow, it becomes 23. So, 8 and 23 that is iteration number 2. So, where the auxiliary flow will go? Auxiliary flow will go along the shortest path. So, just assign the whole 2000 flow there. But the actual flow am I making it 2000? No. That is the link flow update, that is where is the beauty. So, what I will do? I will say let us see how we got this 667, 667 how we can get?

$x_{a,n}$  in this case  $x_{a,n}$  is what just previous value 1000. So, it was 1000 earlier, so  $1000 + 1$  by  $n$  1 by  $n$ ,  $n$  is what? This is iteration 2. So, 1 by 2 into  $Y_{a,n}$  is how much? Here  $2000 - x_{a,n}$  1000. So, you get actually then how much? So, the values are let us say again I explained to you. You had now become it is town flow is auxiliary flow is 2000. So, how we get this second iteration 1000, that I am saying.

So, I am getting it 1000 I am getting it as previous value  $0 + 1$  by 2 into this is the auxiliary flow  $2000 - x_{a,n}$  0. So, half into 2000 eventually so, 1000. And how it becomes 2000? 2000 the calculation is previous value  $2000 + 1$  by 2, the present auxiliary flow  $0 -$  the previous link flow 2000. So,  $2000 - 2000$  by 2, so it becomes 1000. Now with 1000 1000 the cost becomes 18 and 16.5. So, what is the cheaper bypass? So, the auxiliary flow is assigned to bypass.

Now how I get 667? My previous link flow was 1000  $1000 + 1$  by iteration number 3. So, 1 3rd, so  $1000 + 1000 +$  one third of  $0 - 1000$  the auxiliary flow, now is 0. So,  $1000$  mine  $+ one third of 0 - 1000$ . So, that means  $1000 - one third of 1000$ , so that you get 667. How you get 1333? It is  $1000 + one third of 2000 - 1000$ . That is what is shown here in this calculation. So,  $1000 + 1$  third of  $2000 - 1000$ , so that you get 1333.

You can see it start distributing I am putting it here to there, there to here as that shortest path is changing, I am trying to adjust it. So, then accordingly now with 667 flows in town and 1333 flow in bypass we recalculate we find now the travel time becomes 14.67 and 18.67, so

accordingly the auxiliary flow is assigned to town and again you update and like that you proceed. So, multiple iterations may be required, so it is just like that is with us is higher.

Then shift it then again next, we are trying to shift the volume further. So, this link flow update based on consideration of link flow actual link flow in the previous iteration. And the present auxiliary flow using this equation whatever is shown here  $x_{a,n+1}$  by  $n$  equal to  $x_{a,n+1}$  by  $n$  into  $y_{a,n} - x_{a,n}$ . So, in the  $n + 1$  iteration what will be the link flow that will depend on  $n$ th iteration in the sampling whatever was the link flow.

The value of  $n$  and also this difference between the present auxiliary flow and the  $n$ th iteration auxiliary flow and the  $n$ th iteration what was the actual flow. So, that will govern that. So, and we have sent here this auxiliary flow we are calling it as  $Y_{n,a}$  and link flows, we are calling it as this values  $x_{n+1}$ , so  $x_{a,n+1}$ . So, it is the  $n$ th iteration then auxiliary flows are for the  $n$ th iteration and the link flows we are calculating for  $n + 1$  iteration.

That is why it is presented in this tabular form. If you keep on doing like that you see finally, we are getting a flow distribution of 909 and 1091, 909 along the town and 1091 along the bypass. And both cases the travel time at convergence is 17.09. So, it is you will eventually you have converged and converts so nicely, that you are getting both values as you know equal travel time. So, iteration to iteration it is trying to sometimes it may swing you came to this side now you have gone to extreme on the other side.

Then you again will come back it may oscillate. And remember that this  $1$  by  $n$  whatever we said this is generally we can call it a coefficient which any value you can take. You may take even a fixed value rather than making it this  $1$  by  $n$ ,  $1$  can even take a fixed value point  $1.2$  anything any value you can take between  $0$  to  $1$  any value you can take less than  $1$  greater than  $0$ . So, but then it is found that finally the oscillation will reduce whenever you know finally your your value becomes stable both travel time equal.

Then as you are moving towards your solution the oscillation will reduce. So,  $1$  by  $n$  if you use as the multiplier rather than a fixed value making it a function of the number of iterations.

Number of iterations only thing as number of iterations will include the values will become 1 by n the increasingly smaller value. So, I am I know that now my oscillation will reduce. So, one by one if you use as a factor it gives you a much better convergence, that is what is found and that is why we have used it here. But just for your knowledge that its one can use a small value also you can start with the fixed value may be 0.1 right from the beginning.


But then maybe it will take more iterations to converge. So, this you make this factor also a variable and make it a function of iteration that perfectly matches the understanding. That as you are moving more and more iteration wise as we are moving forward, we are actually going towards the convergence. So, the oscillation will reduce that is the basic idea.

**(Refer Slide Time: 38:42)**



### Summary

---

- Capacity Restraint Assignment
  - ✓ Algorithm
  - ✓ Limitation
  - ✓ Modified algorithm for CRA
  - ✓ Example
- Method of Successive Averages
  - ✓ Algorithm
  - ✓ Example



---

  IIT Kharagpur | Urban Transportation Systems Planning | Module F 18

So, we discussed here about the capacity restrained algorithm, the problem with basic capacitive strength algorithm, how to overcome with modified capacity resistant algorithm and then also discussed another heuristic approach called method of successive average told you the algorithm. And explain to you with an example. Thank you so much. So, we shall continue in the next class two more lectures to complete this module F. Thank you so much.