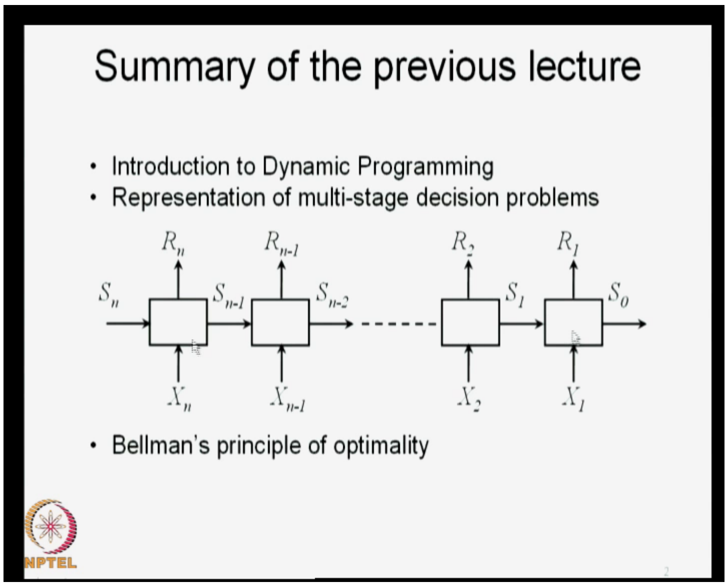
Water Resources Systems

Prof. P. P. Mujumdar
Department of Civil Engineering
Indian Institute of Science, Bangalore

Lecture No # 15
Dynamic Programming Water Allocation Problem

Good morning and welcome to this lecture number 15 of the course, water resource systems modeling techniques and analysis. In the previous lecture, we have introduced dynamic programming. Essentially we started with a representation of multi-stage decision problems, and introduced the Bellman's principle of optimality.

(Refer Slide Time: 00:42)



Summary of the previous lecture

- Introduction to Dynamic Programming
- Representation of multi-stage decision problems

$$\begin{tikzcd} & R_n & & R_{n-1} & & R_2 & & R_1 & \\ S_n \arrow[r] & \boxed{} \arrow[u] \arrow[r, "S_{n-1}"] & & \boxed{} \arrow[u] \arrow[r, "S_{n-2}"] & \cdots \arrow[r] & \boxed{} \arrow[u] \arrow[r, "S_1"] & & \boxed{} \arrow[u] \arrow[r, "S_0"] & {} \\ & X_n \arrow[u] & & X_{n-1} \arrow[u] & & X_2 \arrow[u] & & X_1 \arrow[u] & \end{tikzcd}$$

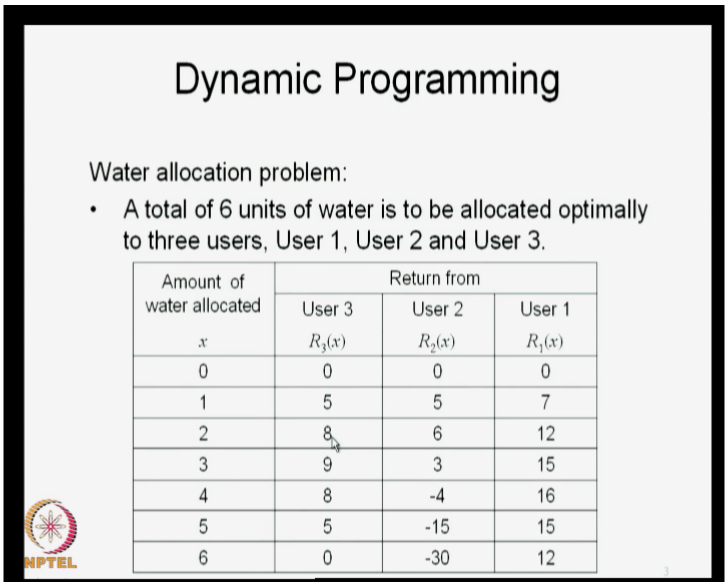
- Bellman's principle of optimality

If you recall, multi-stage decision problem is represented as shown here. The input state at a particular stage n gets transformed to an output state S n minus 1 in this case starting with S n. Because of the decision we make X n and this output state forms an input to the next stage n minus 1, and that gets transformed because of the decision X n minus 1 we make on the state S n minus 1 into S n minus 2, which becomes an input to the next stage and so on. Because of the decision we make here X n at stage n, you get a return of R n.

Now, these returns in water resources problems can be either economic returns, or physical returns in terms of the amount of hydropower produced, in terms of the crop yield that you have accrued, in terms of the flood control benefit that you get, in tangible benefits perhaps and so on. So, the returns are a function of the decisions that you make X n at stage n starting with the state S n at stage n. This is how we represent a multi-stage decision problem, and the optimization we do in dynamic programming, such a multi-stage decision problem is based on the Bellman's principle of optimality.

Essentially, recall that it states starting with a particular state at a particular at a given stage; you must proceed in an optimal manner until all the stages are completed, irrespective of how you arrived at that particular state. That means, as long as you know the state in a particular stage, you proceed in an optimal manner starting with that state. Do not worry about how you arrived at that particular state in that stage. So, given the state at a particular stage, you must proceed in an optimal manner until the end of the horizon. This horizon can be space; it can be time and so on.

(Refer Slide Time: 02:53)



Dynamic Programming

Water allocation problem:

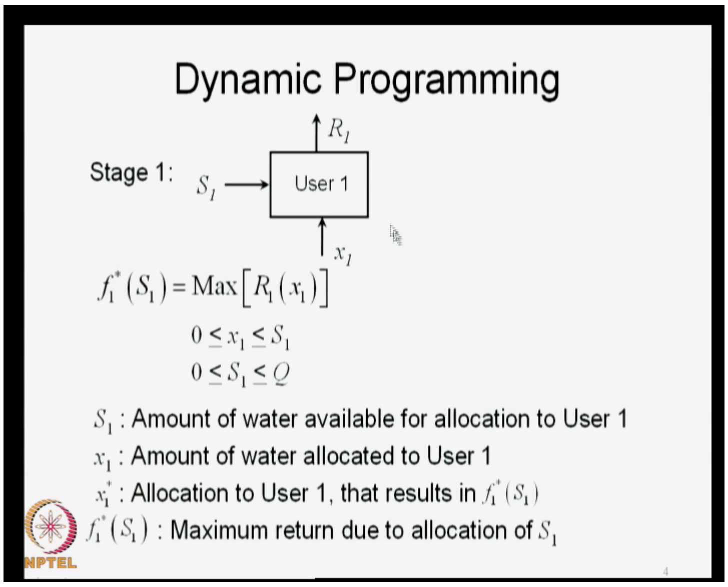
- A total of 6 units of water is to be allocated optimally to three users, User 1, User 2 and User 3.

Amount of water allocated	Return from		
	User 3	User 2	User 1
x	$R_3(x)$	$R_2(x)$	$R_1(x)$
0	0	0	0
1	5	5	7
2	8	6	12
3	9	3	15
4	8	-4	16
5	5	-15	15
6	0	-30	12

We started discussing the water allocation problem. For completeness sake, I will restate the problem and then, the stage one computations that we did in the previous lecture, we will just go through it again quickly. There are three users. An amount of water available of 6 units needs to be allocated optimally among the three users. The return that you get from a particular user for a given allocation is given in this table. For example, if you allocate 2 units of water to user number 3, you get a return of 8 units. If you allocate an amount of 2 units to user two, you get a return of 6 units, allocate an amount of 2 units to user 1, you get a return of 12 units and so on.

So, this table gives the returns that you get for a particular allocation. Look at this. If you allocate 4 units of water to user number 2, you get a negative benefit. Typically, this can be an irrigation user, which means up to certain point; you keep on increasing the benefits and then, beyond that it starts decreasing. In fact, it may become negative, in the sense that if you allocate more water than actually needed, you may suffer losses in fact. So, this is the type of benefit functions that you you may get. Now, these need not be always economic returns. They can be intangible benefits; they can be benefits in terms of physical quantities and so on.

(Refer Slide Time: 04:41)



Dynamic Programming

Stage 1:

$$\begin{tikzcd} S_1 \arrow[r] & \boxed{\text{User 1}} \arrow[u, "R_1"] \\ & x_1 \arrow[u] \end{tikzcd}$$

$$f_1^*(S_1) = \text{Max}\left[R_1(x_1)\right]$$

$$0 \le x_1 \le S_1$$

$$0 \le S_1 \le Q$$

S_1 : Amount of water available for allocation to User 1

x_1 : Amount of water allocated to User 1

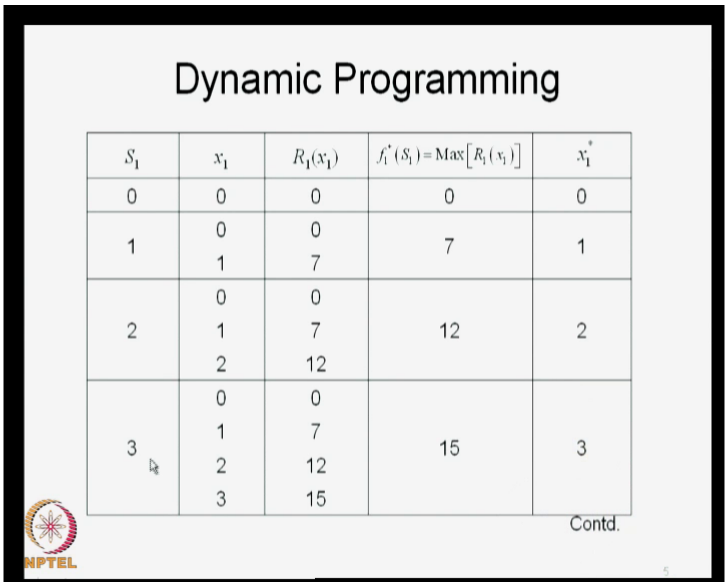
x_1^* : Allocation to User 1, that results in $f_1^*(S_1)$

$f_1^*(S_1)$: Maximum return due to allocation of S_1

So, we started this example from stage Stage number 1, where we include only 1 user, user number 1. We say that, S 1 which is the state at stage number 1 is the amount of water available to be allocated to user number 1, and S 1 can vary from 0 to Q, where Q is the total amount of water available. In this case, it is 6 units. So, we solve this problem for all possible values of S 1 because we do not know what exactly S 1 will be. We are moving in the backward direction. So, we are asking the question, if S 1 is equal to 0, how much I should allocate, if S 1 is equal to 1, how much I should allocate, if S 1 is equal to 2, how much I should allocate and so on.

So, we are solving the problem for a known, for a given value of S 1, which is not known. We will know the actual value of S 1, only when we solve the complete problem, and f 1 star S 1 is the optimized objective function value for a given S 1 at that stage 1, where only 1 user is included. So, this is how we solve. We say S 1 is equal to 0, X 1 can be only 0. Recall here that X 1 can take on values up to S 1. What we mean by that is, that if 3 units of water available, you cannot allocate anything more than 3 and therefore, X 1 can go on to take on values between 0 and S 1.

(Refer Slide Time: 06:15)



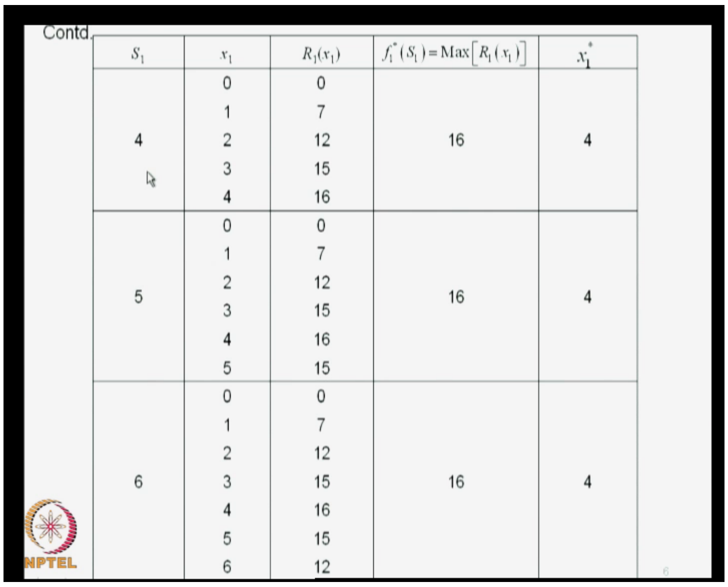
Dynamic Programming

S_1	x_1	$R_1(x_1)$	$f_1^*(S_1) = \text{Max}\left[R_1(x_1)\right]$	x_1^*
0	0	0	0	0
1	0	0	7	1
	1	7		
2	0	0	12	2
	1	7		
	2	12		
3	0	0	15	3
	1	7		
	2	12		
	3	15		

Contd.

So, we ask the question if S 1 is 1, whether we should allocate 0 or we should allocate 1, and the maximum benefits are 7 corresponding to an allocation of 1. Similarly, if S 1 is equal to 2, you can either allocate 0 or 1 or 2. If you allocate 0, you get a return of 0, if you allocate 1, you get a return of 7, if you allocate 2, you get a return of 12 and therefore, the maximum benefit is, maximum return is 12. Therefore, the corresponding allocation is 2.

(Refer Slide Time: 07:00)

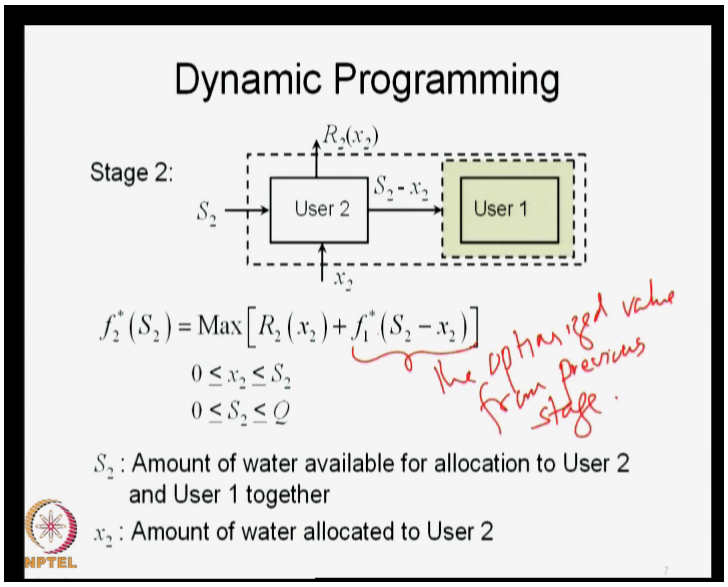


Contd.

S_1	x_1	$R_1(x_1)$	$f_1^*(S_1) = \text{Max}[R_1(x_1)]$	x_1^*
4	0	0	16	4
	1	7		
	2	12		
	3	15		
	4	16		
5	0	0	16	4
	1	7		
	2	12		
	3	15		
	4	16		
	5	15		
6	0	0	16	4
	1	7		
	2	12		
	3	15		
	4	16		
	5	15		
	6	12		

So, what it states is, that if 2 is available, allocate 2 and you get a return of 12. If 3 is available, allocate 3 and you get a return of 50. If 4 is available, allocate 4 and you get a return of 16. If 5 is available, allocate 4 and you get a return of 16. Remember, if 5 is available, if you allocate all the 5, then you get a return of 50, whereas if you allocate only 4, you get a return of 16. Therefore, it says, if 5 is available, allocate only 4. Similarly, if 6 is available, allocate only 4. I encourage you to look at the previous lecture; that is a lecture number 14 for a detailed discussion on this.

(Refer Slide Time: 07:34)



Dynamic Programming

Stage 2:

$R_2(x_2)$
S_2
User 2
$S_2 - x_2$
User 1
x_2

$$f_2^*(S_2) = \text{Max}\left[R_2(x_2) + \underbrace{f_1^*(S_2 - x_2)}_{\text{the optimized value from previous stage.}}\right]$$

$$0 \le x_2 \le S_2$$
$$0 \le S_2 \le Q$$

S_2 : Amount of water available for allocation to User 2 and User 1 together

x_2 : Amount of water allocated to User 2

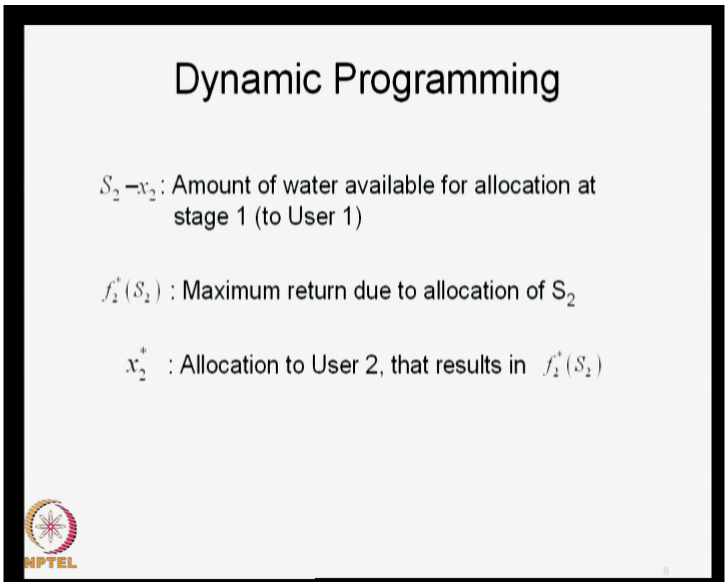
We will now proceed to stage number 2. In stage number 2, we are including 2 users now, user number 1 and user number 2 together. So, S 2 which is the state at stage number 2 is the amount of water available to be allocated to user number 1, and user number 2 together, and X 2 is a decision that we are making at stage 2. This is the amount of water allocated to user number 2 because you make an allocation of X 2 from the available S 2, the amount of water available for allocation at the next stage, which is S 1 in this case as we are proceeding the backward direction, will be S 2 minus X 2, but we have to solve this example, solve this problem. Already, for all amounts of water available here, we have solved the problem in the stage number 1 and therefore, we know what the optimal benefit that you get out of an allocation of S 2 minus X 2 is, and that is what we pick up. So, we write at stage 2. This recursive relationship you understand correctly.

At stage 2, the optimal benefit associated with the state S 2 is given by maximum value of the returns that you get from user 2 for an allocation of X 2, which is R 2 of X 2 plus the optimized results, optimized return that you get for an allocation of the remaining amount of water, which is S 2 minus X 2 in the previous stage or the next stage in this particular case to user number 1. So, what does this indicate? This term indicates here the optimized value from previous stage.

Remember, this is what we have solved already from the previous stage computations. This is what we have solved f 1 star S 1, which means given this argument S 1 here; we know what the X 1 is star and what the maximum allocation is. Here, Maximum objective function value, and that is what we use here. I repeat again, given the state which is the amount of water available at stage 2 to be allocated to both user number 1 and user number 2 together, we make a decision on X 2, which is the amount of water allocated to user number 2, we get a return of R 2, X 2. Because of this decision, we are left with S 2 minus X 2 for the next user, and we have already solved in the previous stage, what should be the optimal value corresponding to S 2 minus X 2, and that is what we add here. So, this recursive relationship is in fact a statement of the Bellman's principles of optimality.

Given the state S 2, at this particular stage, we are proceeding in an optimal manner until the end of all the stages, until all the stages are included. So, we are getting the maximum value here plus as the result of which, as the result of this allocation, whatever state that we end up in, that becomes an input state and for which we have already solved, and that is optimized objective function value that we use. Again, S 2 which is the amount of water available at stage 2 can take on only values up to Q, which is the maximum available water, which in this particular case is 6 units. So, we solve this problem for S 2 taking on values 0, 1, 2, 3, 4 etcetera up to 6, and X 2 which is the amount of water allocated at for user 2 can go on up to S 2. Let us say, S 2 is 4 units, then X 2 which is the amount of water allocated cannot be anything more than 4 units. Therefore, X 2 goes on up to S 2.

(Refer Slide Time: 12:26)



Dynamic Programming

$S_2 - x_2$: Amount of water available for allocation at stage 1 (to User 1)

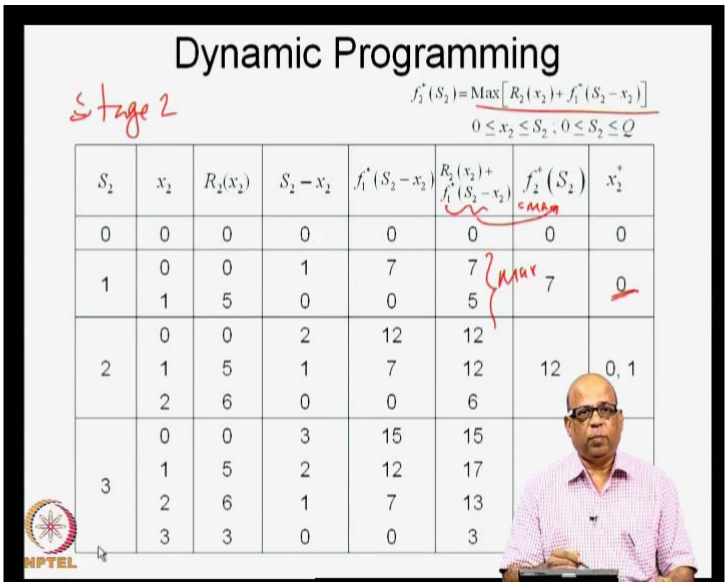
$f_2^*(S_2)$: Maximum return due to allocation of S_2

x_2^* : Allocation to User 2, that results in $f_2^*(S_2)$

Now, stage 2 computations. Computation is the big word for this. Simple calculation is what you understand correctly. Then, any number of stages, you can formulate for a given problem. So, S 2 minus X 2, as I said will be the amount of water available for allocation at stage 1 and in stage 1, we have only 1 user. So, that is to user 1 and f 2 star S 2, this one, this term here is the maximum return due to the allocation of S 2, which means together to user 1 and user 2.

What is the optimum benefit or optimum return that you get from such an allocation and X 2 star is allocation to user 2. That maximizes, that corresponds to f 2 star X S 2. There are lots, there are large number of possible values of X 2, out of which one particular or some of them will lead to f 2 star S 2, and that allocation, we denote by X 2 star. So, this is the notation, the DP problems, especially the discrete dynamic programming problems, where the variables, state variables as well as the decision variables take on only discrete values or typically done through computations.

(Refer Slide Time: 13:45)



Dynamic Programming

Stage 2

$$f_2^*(S_2) = \text{Max}\left[\underline{R_2(x_2) + f_1^*(S_2 - x_2)}\right]$$

$$0 \le x_2 \le S_2 \,;\, 0 \le S_2 \le Q$$

S_2	x_2	$R_2(x_2)$	$S_2 - x_2$	$f_1^*(S_2 - x_2)$	$R_2(x_2) + f_1^*(S_2 - x_2)$	$f_2^*(S_2)$ <Max	x_2^*
0	0	0	0	0	0	0	0
1	0	0	1	7	7	7	0
	1	5	0	0	5		
2	0	0	2	12	12	12	0, 1
	1	5	1	7	12		
	2	6	0	0	6		
3	0	0	3	15	15		
	1	5	2	12	17		
	2	6	1	7	13		
	3	3	0	0	3		

So, let us look at how we write this for stage number 2. For stage number 1, we have completed the computations. So, we will start with this is for stage 2. Typically, this is the way we write every table. You write the stage number here. So, stage 2 and then, you write the recursive relationship. This is called as a recursive relationship, which gives you for a given state. In that stage, it gives an expression for the immediate objective function value because of the decision that you make here plus the optimized objective function value for all the remaining stages together.

So, f 1 star S 2 minus X 2, X 2 is the optimized objective function value that you have obtained in the previous stage of calculations. So, look at this now. S 2 is 0; that means we are saying if S 2 is 0, what should be my X 2. X 2 cannot be anything other than 0 because 0 X 2 takes on values between 0 and S 2. So, S X 2 has to be 0. If X 2 is 0, R 2 of X 2 is 0. This is the return that we get from an allocation of 0 units to user number 2, that is, R 2 of X 2 is what we are writing here and then, S 2 minus X 2. So, 0 minus 0, that is again 0, and f 1 star corresponding to S 2 minus X 2, this is picked up from the previous stage calculations.

So, the previous stage calculations we have corresponding to S 1 is equal to 0, f 1 star of S 1 is 0. So, we pick up 0 there. This is 0, and if in the case of 0 everything becomes 0, so it says that if 0 is available, allocate 0 and you get a maximum return of 0. We look at 1 and 2 and then, continue. If 1 is available, that means what we are saying is S 2 is equal to 1, you can either allocate 0 or allocate 1 at X 2. If you allocate 0, you get a return of 0 from X 2, from the allocation X 2. If you allocate 1, you get a return of 5. If 1 is available, and you allocated 0 to user number 2, how much are we left for the user number 1? 1 minus 0, which is 1, that is S 2 minus X 2 and if 1 is available, you know what has happened in the previous case, previous stage corresponding to 1. It says that, if 1 is available here, you allocate 1 and you will get a return of 7. So, this is what we write here. So, this is 7 here, 1 0 1. This 7 here is got from the previous stage calculations.

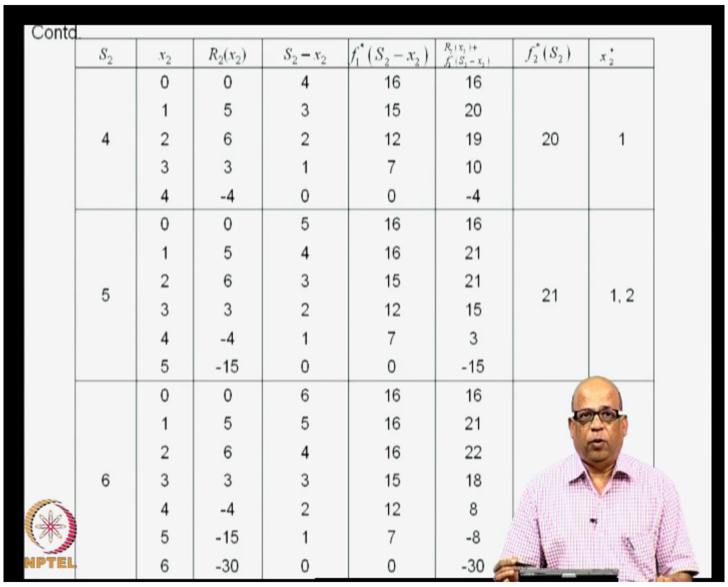
So, if S 2 is 1 and you allocated 0 to user number 2, you are left with 1 for the user number 1 which gives you optimized benefit of 7, optimized return of 7. So, the total return corresponding to this combination 0 and 1 will be 7, that is this 0 plus this 7 that is equal to 7. So, R 2 X 2 plus f 1 star S 2 minus X 2 that would be 7. Starting with S 2 equal to 1 if you allocate X 2 as 1, that means, X 2 is equal to 1, you get a return of 5. Then, you how much you are left with? 1 minus 1, that is 0, you are left with 0.

Corresponding to 0, the previous Stage calculations show that f 1 star of 0 is 0. So, you get a total benefit of 5 plus 0, which is 5 and therefore, 7 is the maximum value among these. So, this is you pick up a maximum among these. So, this will be equal to maximum of this term. That is what we have written here. So, this is 7, and it says that, that corresponds to 0. So, X 2 star is 0 here. What does it say? It says that, at the beginning stage number 2, if you are given an amount of water of 1, then you allocate 0 to user number 2 is what it says. So, star is 0.

You look at 2, if 2 is available; you can either allocate 0, R 1, R 2 to user number 2. If you allocate 0, you get a return of 0 from user 2. If you allocate 1, you get a return of 5 from user 2. If you allocate 2, you get a return of 2 from user 2. If you allocate 0, you are left with 2 for the user number 1, and the optimal allocation of 2 gives you 12 from user number 1. Look look at this. User number 1, you have already solved and you are looking at allocation of 2 at user number 2. That gives you a benefit of 12. So, you pick up that f 1 star corresponding to this, a value of 2 and that is what you get 12.

Similarly, if you allocate 1 to user number 2, how much you are left for the user number 1? 2 minus 1, that is 1 and if you have 1 left there, the optimized value is 7. Optimized value corresponding to that stage, which is stage number 1, is 7. Similarly, if you allocate 2 here when you are left with 0 here, 2 minus 2 is 0, you get are return optimized value of 0 here. So, what is the total that you get? 12 plus 0, that is 12, 7 plus 5 that is 12, 0 plus 6 that is 6, and pick up the maximum of that, maximum is 12. Therefore, it says that if 2 is available, you either allocate 0 or 1, both of which gives you a value of 12. Like this for all the possible discrete values of S 2, you solve this problem.

(Refer Slide Time: 20:47)



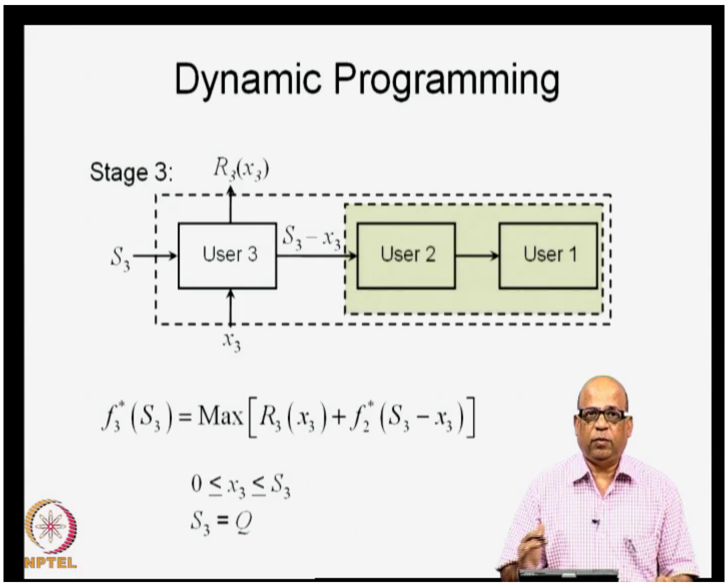
Contd.

S_2	x_2	$R_2(x_2)$	$S_2 - x_2$	$f_1^*(S_2 - x_2)$	$R_2(x_2) + f_1^*(S_2 - x_2)$	$f_2^*(S_2)$	x_2^*
4	0	0	4	16	16	20	1
	1	5	3	15	20		
	2	6	2	12	19		
	3	3	1	7	10		
	4	-4	0	0	-4		
5	0	0	5	16	16	21	1, 2
	1	5	4	16	21		
	2	6	3	15	21		
	3	3	2	12	15		
	4	-4	1	7	3		
	5	-15	0	0	-15		
6	0	0	6	16	16		
	1	5	5	16	21		
	2	6	4	16	22		
	3	3	3	15	18		
	4	-4	2	12	8		
	5	-15	1	7	-8		
	6	-30	0	0	-30		

Then, let us say you go to 5, for example 0 1 2 3 4 5. If X 2 is 0 1 2 3 4 5, then you will get corresponding R 2 value. Then, this will be 5 minus 0, 5 minus 1, 5 minus 2 etcetera and then, pick up the associated values of optimized returns from the previous stage, they will be 16, 16 etcetera like this. So, the maximum value will be 21 out of all these allocations and 21, it says that if 5 is available, you either allocate 1 or 2 to user number 2 and you get a maximum benefit, maximum return of 21. So, like this, you continue till you exhaust all possible values of S 2.

What are all the possible values of S 2? S 2 can be either 0 or 1 or 2 etcetera up to 6. So, you complete for all the 6. This completes the calculations for stage number 2 and then, you proceed to stage number 3. So, remember in stage number 2, what do we answer? We answer the question, that given S 2 which is the amount of water available to be allocated to both user number 1 and user number 2 together. How much should I allocate to X 2, such that the total return obtained from the immediate allocation X 2 to user number 2 plus the optimized return out of the resultant state S 2 minus X 2 at stage number 1, these 2 together will be optimized? So, we are moving in an optimal manner until the end of the horizon, which includes user number 1. So, this is what we answered at stage number 2.

(Refer Slide Time: 22:47)



Dynamic Programming

Stage 3:

$$f_3^*(S_3) = \text{Max}\left[R_3(x_3) + f_2^*(S_3 - x_3)\right]$$

$$0 \le x_3 \le S_3$$
$$S_3 = Q$$

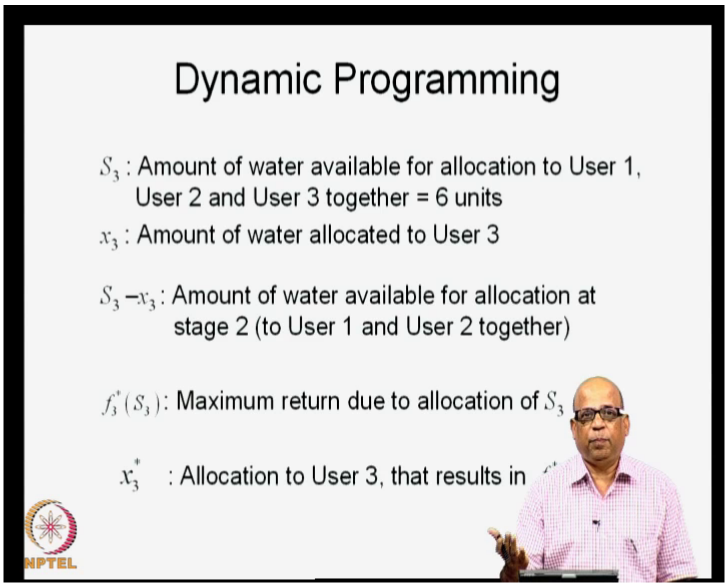
Now, we go to the next stage, which is stage number 3. Remember in stage number 3, we add 1 more user here, user number 3. So, we have already solved at stage 2, we have solved for the amount of water available S 2 here. We know how much optimal benefit that we get. So, at stage 3, we are asking the question if S 3 is available for allocation to user 3, user 2 and user 1 together. How much should I allocate to user number 3, such that the total returns that you get out of this full system is maximized, and that is what we write it as a recursive relationship f 3 star of for a given value of S 3 is equal to maximize R 3 of X 3.

You are allocating a value amount of X 3 to user number 3. As a result of which, you get a return of R 3 of X 3 plus because you allocated X 3 from an available S 3, you are left with S 3 minus X 3 to be allocated to user number 1 and user number 2 together. This problem you have solved in the previous stage. In the previous stage, essentially you solve for user number 1 and user number 2 for a given value of S 2 and this value of S 2 comes from S 3 minus X 3, and that is the term that we write here. So, this term here f 2 star S 3 minus X 2, S 3 minus X 3 is in fact, the optimized value of the objective function corresponding to the value S 3 minus X 3, which we have solved in the previous stage.

So, you pick up from the previous stage the associated value S 3 minus X 3, but there is some interesting feature here. In stage number 2, what did we do? In stage number 2, we said we will solve this problem for all values of S 2 going from 0 to Q. That is why we solved it from 0 for given value 0, 1, 2, 3 etcetera up to 6. When you come to Stage 3, in this particular example, all the users are included, and the state variable that is the state of the system S 3 is the amount of water available to be allocated to the users included in that stage, and in stage 3, all the users are included. Therefore, S 3 will be equal to the total amount of water available. S 3 is Q, so S 3 will not take any values other than the total amount of water available, which is 6.

So, when you come to the last user in water allocation problem, such as what I am discussing now. When you come to the last user, that is the last stage, the total amount of water available is, it becomes the state of the system. So, S 3 will be Q. So, you do not solve for various values of S 3, but the given value which is the total amount of water available, 6 in this particular case. S 3 is equal to Q and then, X 3 goes on from, X 3 can take on any value between 0 and S 3, which is equal to Q in this case. So, this is what we solve in the last stage.

(Refer Slide Time: 26:27)



Dynamic Programming

S_3 : Amount of water available for allocation to User 1, User 2 and User 3 together = 6 units

x_3 : Amount of water allocated to User 3

$S_3 - x_3$: Amount of water available for allocation at stage 2 (to User 1 and User 2 together)

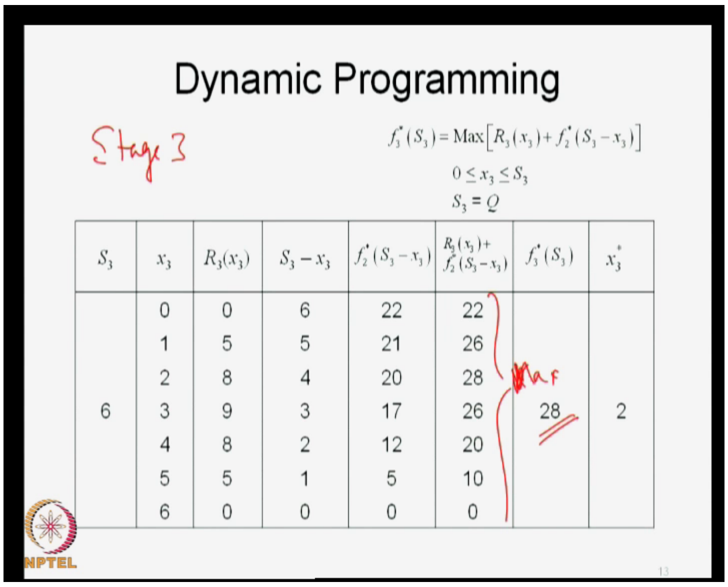
$f_3^*(S_3)$: Maximum return due to allocation of S_3

x_3^* : Allocation to User 3, that results in

So, let us see what we do in the last stage. So, this is what I mentioned. S 3 will be 6 units. X 3 is the amount of water allocated to user 3. S 3 minus X 3 is the amount of water available for allocation at stage 2 and stage 2 includes 2 users, user 1 and user 2. X

3 star is allocation to user 3 that results in f 3 star S 3, much the same way as we did for stage number 2. However, remember here that S 3 minus X 3 is available for user number one and user number two together. You do not again go to stage 1 now. You simply relate it with what you have solved for this stage 2 because in stage 2 computations, you have already included the optimized value of stage 1. So, when you are doing for stage 3 simply look at the previous table and then, pick up the optimal values corresponding to a given value of S 3 minus X 3.

(Refer Slide Time: 27:32)



Dynamic Programming

Stage 3

$$f_3^*(S_3) = \text{Max}\left[R_3(x_3) + f_2^*(S_3 - x_3)\right]$$
$$0 \le x_3 \le S_3$$
$$S_3 = Q$$

S_3	x_3	$R_3(x_3)$	$S_3 - x_3$	$f_2^*(S_3 - x_3)$	$R_3(x_3)+$ $f_2^*(S_3 - x_3)$	$f_3^*(S_3)$	x_3^*
	0	0	6	22	22		
	1	5	5	21	26		
	2	8	4	20	28	Max	
6	3	9	3	17	26	28	2
	4	8	2	12	20		
	5	5	1	5	10		
	6	0	0	0	0		

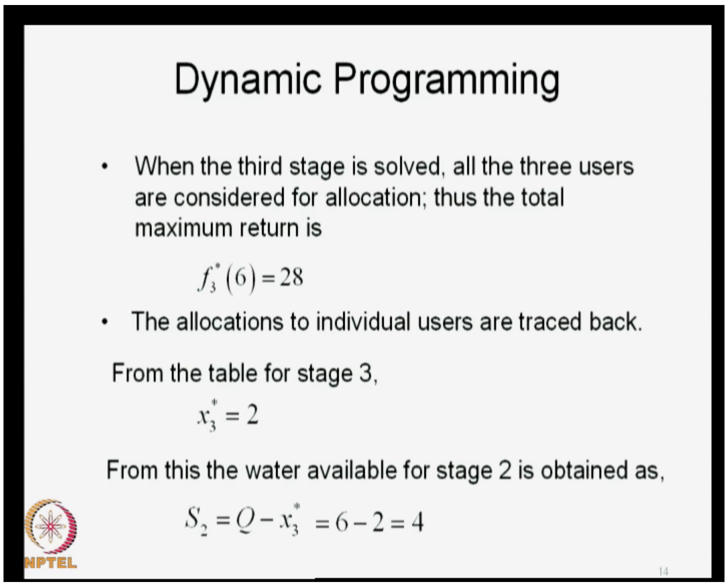
So, this again we do in the tabular form. S 3 now, takes on exactly 1 value 6 because it is a total amount of water available to be allocated to all the users included in stage 3, and in stage 3 now, we have included user 1, user 2, and user 3 together. In stage 2, we have already solved for user 1 and user 2 together. So, we ask the question, if S 3 is available, then, how much should I allocate to user number 3? Should I allocate 0, 1, 2, 3, 4 etcetera, such that the total return that you get is maximized. So, I get a return, immediate return of R 3, X 3 plus because I have allocated X 3 from the available S 3, I am left with S 3 minus X 3 for the next stage, which includes 2 users.

I know how much I get from an available, availability of S 3 minus X 3 at the next stage because this is exactly the problem that we have solved in the previous stage. So, this is the recursive relationship. So, we do exactly the same way as we did for the previous stage. We say if 6 is available, I can either allocate 0 to X 3 or 1 or 2 etcetera. If I allocate 0, I get a return of 0 from user 3, return of 1 from user 3, return of 8 from user 3 and so on. Out of 6, if I look at 0, I am left with 6 for the other 2 users and the other 2 users corresponding to 6, I have already solved corresponding to 6 the optimal value is 22 here, I pick up this value 22 and that is we take sum of these 2, that is, this term sum that is 22. Look at this 6. I allocate 1 and therefore, I am left with 5. If I allocate 1, I am getting a return of 5 where is this available, this data is available right at the beginning of the problem. So, if I allocate 1 from user 3, I am getting a return of 5. So, this is the table that I am using there. So, if I allocate 1, I am left with 5, and that 5 from the previous table, you go to 5 and look at the 2 star of 5, that is equal to 21 and that is what you write here 21 and therefore, 21 plus this 5 is equal to 26.

Similarly, if I allocate 2, I am left with 4 and this 4 corresponds to 20 and therefore, the total is 20 plus 8 that is equal to 28 and so on. So, like this, you do it for all the possible values of X 3, and you pick up the maximum value out of all these 3, all these allocations. So, the maximum value in this case corresponds to 28. Maximum value is 28 and the particular X star, particular X 3 value that results in this maximum value is 2. What I what I mean by that is, it says that if 6 is available, I am sorry 6 is available at stage 3, then you allocate 2 to user 3 and you get a maximum benefit of 28. This is in fact the maximized value of the objective function after you complete all the allocations.

Remember you should not add this to the value that you got in the previous table associated with any particular allocation and so on because this value that you get has already included the optimized value for the stage 2, which has already included the optimized value for stage 1. Therefore, when you solve for the last stage in this particular case, stage number 3, this is stage 3. When you solve for the last stage, the objective function value that you get is in fact, the optimized objective function value for the entire problem. Now, from this we need to trace that, though allocations to different users, the last table here gives you an allocation of 2 units to user number 3. It says that you allocate 2 units to user number 3. So, understand how we trace back the solution now.

(Refer Slide Time: 32:27)



Dynamic Programming

- When the third stage is solved, all the three users are considered for allocation; thus the total maximum return is

$$f_3^*(6) = 28$$

- The allocations to individual users are traced back.

From the table for stage 3,

$$x_3^* = 2$$

From this the water available for stage 2 is obtained as,

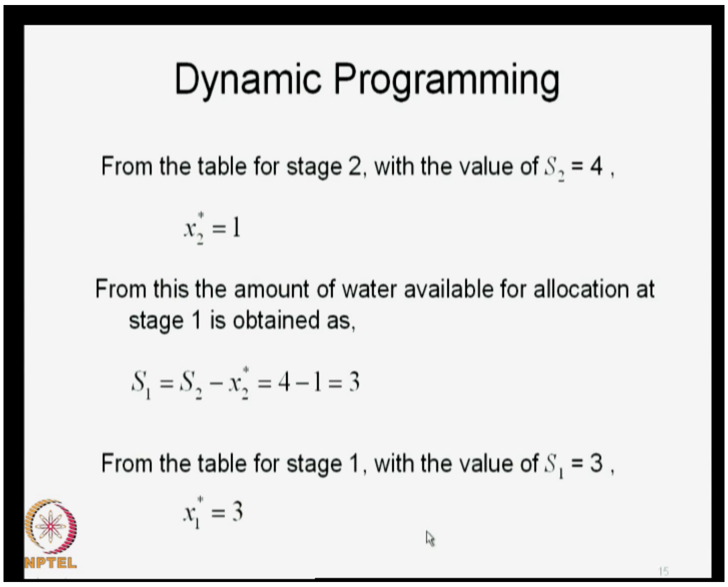
$$S_2 = Q - x_3^* = 6 - 2 = 4$$

14

So, when the third stage is solved, in this particular case, there are only 3 stages. All the 3 users are considered for allocation. Thus, the total maximum return is 28, which means f 3 star of 6, which is the total amount of water available is equal to 28. Now, starting with this, we need to trace back. So, just understand how we trace back. 6 is available further for allocation to all the 3 users and this table says, allocate 2 to user number 3. So, how much are we left? 6 minus 2 which is equal to 4, so 4 units to be allocated to the other 2 users together, that is user number 1 and user number 2 together.

So, we get S 2, which is the amount of water available to be allocated to user number 2 and user number 1 together is Q minus X 3 star, which is equal to 4 units. Then, you go to stage number 2 calculations. So, you are saying that 4 is available. So, S 2 is equal to 4, go to stage number 2 calculation; look at what is the corresponding allocation for S 2 is equal to 4. It says that, if 4 is available, you allocate 1 to user number 2. So, from the previous table, you pick up the optimal allocation corresponding to S 2 is equal to 4, which says that X 2 star is equal to 1. So, from here, you got 6 minus 2 is equal to 4 available for the previous stage and from the previous stage calculation, it says X 2 star is equal to 1.

(Refer Slide Time: 34:16)



Dynamic Programming

From the table for stage 2, with the value of S_2 = 4 ,

$$x_2^* = 1$$

From this the amount of water available for allocation at stage 1 is obtained as,

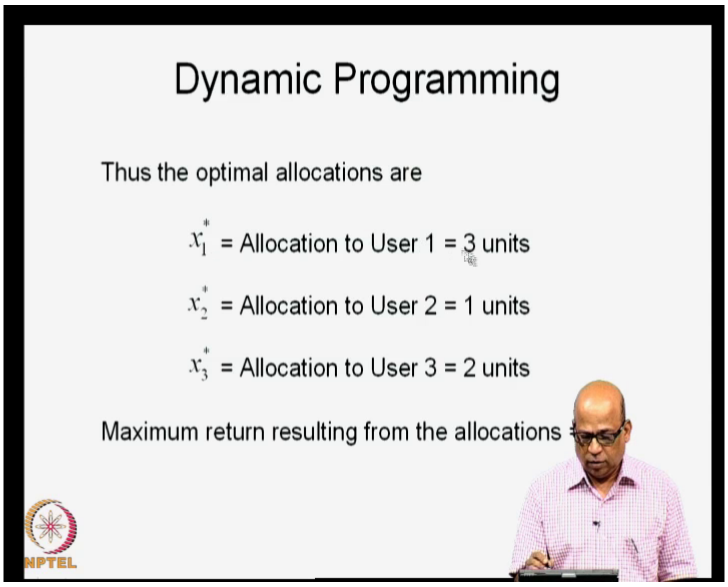
$$S_1 = S_2 - x_2^* = 4 - 1 = 3$$

From the table for stage 1, with the value of S_1 = 3 ,

$$x_1^* = 3$$

So, 4 minus 1 that will be available at stage number 1, that is S 2 minus X 2 star and that will be 3. With this 3, you go back to stage number 1 calculation and look at what you should go for S 1 is equal to 3. It says if S 1 is equal to 3, your X 1 star must be equal to 3. So, this is how you trace back the allocations to all the users. So, you got from here X 3 is equal to 2, X 3 star is equal to 2, then you went to stage number 2 calculations. You got X 2 star is equal to 1, then you went to stage number 1 calculation and you got X 1 star is equal to 3. So, you got the optimal allocations to user number 1, user number 2 and user number 3 together.

(Refer Slide Time: 35:16)



Dynamic Programming

Thus the optimal allocations are

x_1^* = Allocation to User 1 = 3 units

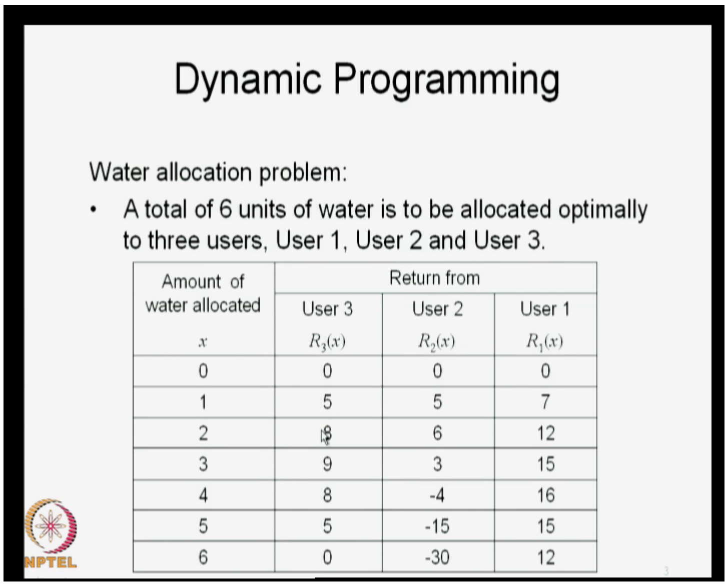
x_2^* = Allocation to User 2 = 1 units

x_3^* = Allocation to User 3 = 2 units

Maximum return resulting from the allocations

The optimal return that you get from such an allocation is 28 units. So, X 1 star is 3, X 2 star is 1, X 3 star is 2 and the total return is 28, total maximum return is 28. These we can verify from the original table of data.

(Refer Slide Time: 35:39)



Dynamic Programming

Water allocation problem:

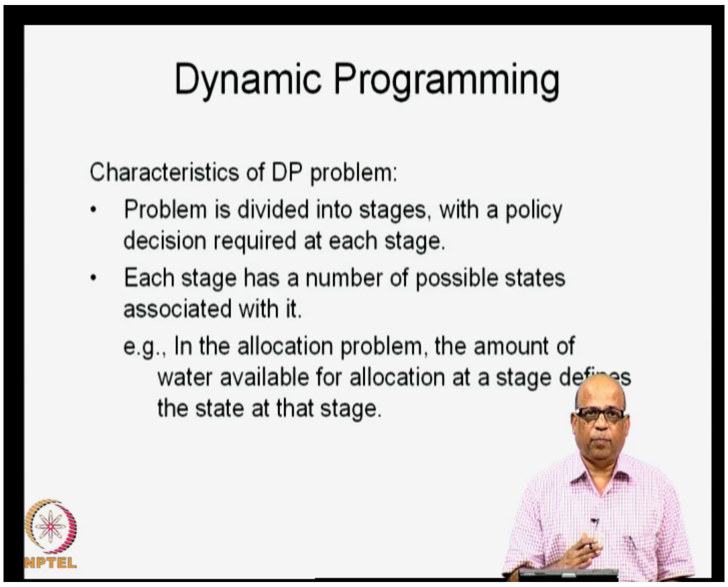
- A total of 6 units of water is to be allocated optimally to three users, User 1, User 2 and User 3.

Amount of water allocated	Return from		
	User 3	User 2	User 1
x	$R_3(x)$	$R_2(x)$	$R_1(x)$
0	0	0	0
1	5	5	7
2	8	6	12
3	9	3	15
4	8	-4	16
5	5	-15	15
6	0	-30	12

Now, this is the data where say, X 3 star is, I think we got X 3 star to be 2 here and let us write down X 3 star is 2, X 2 star is 1, and X 1 star is 3, so 3 1 2, 3 1 2, user 1, user 2 and user 3. So, we will go to that table. 3 1 2 is the allocation. 3 is 15, 1 is 5 and 2 is 8. So, 15 plus 5 plus 8, that is 28. That is the maximum return that you get. Simply look at this table. If you wanted to get to these particular allocations, what you have to do is, you will have to exhaustively enumerate. For example, 6 is available. You may say, 0 6 0, then 0 4 2, 0 2 4 and so on. Like this exhaustively you have to enumerate, but the Bellman’s principle of optimality makes such an optimization very simple because at every stage, we are only looking at the computations of the previous stage.

So, every time you look at, you relate that present stage calculations with what we have done in the previous stage, and that is how the computations becomes simple. So, this is the way we trace back the solutions after you solve the problem for including all the stages. So, you included all the users and then, solved at the last stage. You solve the problem for the total amount of water available from which you get the allocation to be made to that last user that was included in this particular case user 3. Then, you will go to the next stage of computation knowing that from an available amount of water Q, you have allocated X 3 star. Therefore, for the remaining users which are all included in the previous stage, you have an amount of Q minus X 3 available and go to the previous stage computations, pick up what you should do corresponding to this amount of water available and so on. So, like this, you keep tracing back the solution.

(Refer Slide Time: 38:05)



Dynamic Programming

Characteristics of DP problem:

- Problem is divided into stages, with a policy decision required at each stage.
- Each stage has a number of possible states associated with it.

  e.g., In the allocation problem, the amount of water available for allocation at a stage defines the state at that stage.

So, let us look at what we have done in this simple discrete dynamic programming problem. You have a given problem let us say, n variable problem. You divided that into different stages. So, n variable problem you have now divided into n stages, each of which having only 1 variable to solve. In this particular case of water is a user allocation problem that I just solved. You had 3 users, therefore 3 allocations, X 1, X 2 and X 3. If you have formulated a, let us say, it was possible for you to formulate LP problem, then it would have been a 3 variable problem. This you divided into 3 different stages, stage number 1, stage number 2 and stage number 3.

At each stage, you solve for exactly 1 variable. In the first stage, you solve for X 1. In the second stage, you solve for X 2 and from the X 2, arising out of X 2, you know what the total return is. At stage 3, you only solve for X 3. So, at each stage, you are solving for exactly 1 variable. So, what was the 3 variables problem? You converted into a single variable problem to be solved 3 times. Every stage, at each stage, you have a policy decision. What are the policy decision in our allocation problem? The amount of water available to be allocated at that particular stage is the policy decision.

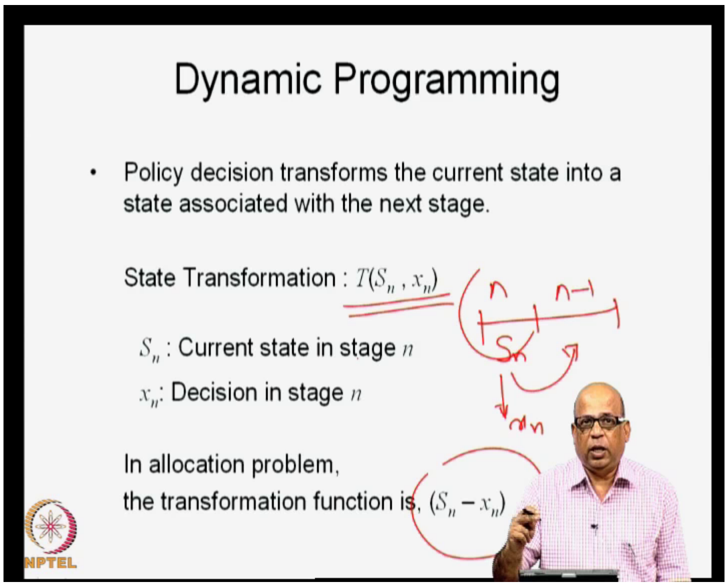
Similarly, at every stage, you have a state of the system at that stage. When we had user number 1, we defined S 1 as the state. As the state of the system at that particular stage, this example is the amount of water available to be allocated to all the users included in that stage. Let say S 1. S 1 is the amount of water available to be allocated to user 1, S 2 is the amount of water available at stage number 2 to be allocated to all the users included in that stage in this particular case will be user number 1and user number 2 together.

Similarly, S 3 is the amount of water available to all the users included in that stage, which is user number 1, user number 2 and user number 3 together. So, at every stage, you have a state variable and in the user, in the water allocation problem, the state variable simply corresponds to the amount of water available. So, the main characteristics of the dynamic programming are that n variable problem has been decomposed into n stage problem. Each Stage you are solving for only 1 variable in this particular case and 1 variable can be 1 vector, one decision vector and at every stage, you have a state of the system and at every stage, you make a policy decision.

The policy decision in the water allocation problem is how much to be allocated to that particular user. For stage number 2, you are making a decision on X 2, in stage number 3, you are making a decision on X 3 and stage number 1; you are making a decision on X 1 and so on. So, at every stage you have a policy decision, at every stage you have a state variable which completely describes the state of the system, at every stage you have an immediate return that comes out of that. So, these are the characteristics of the dynamic programming.

The articulation of the problems to fit the dynamic programming algorithm or dynamic programming structure is in fact an important aspect of dynamic programming. As I mentioned in the last class, a given problem you need to look at what are the stages can we look into different stages and can we identify the state of the system, can we completely describe the state of the system at every stage, can we identify what is the policy decision and so on.

(Refer Slide Time: 42:57)



Dynamic Programming

- Policy decision transforms the current state into a state associated with the next stage.

State Transformation : $T(S_n, x_n)$

S_n : Current state in stage n

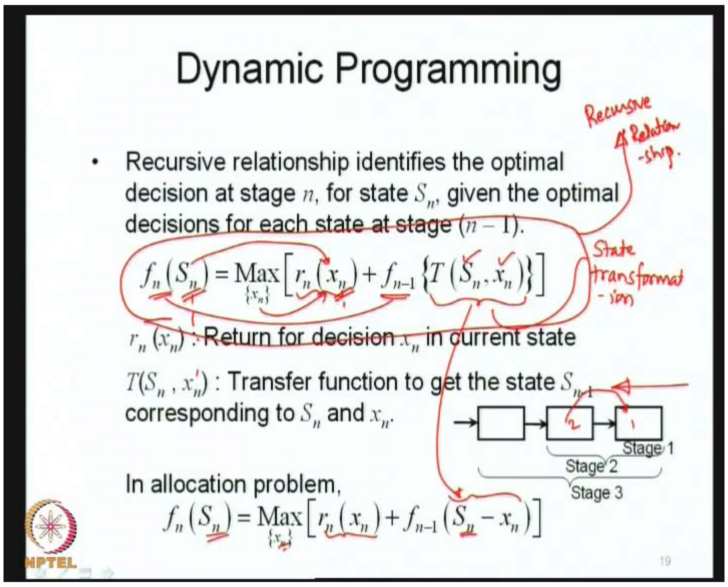
x_n: Decision in stage n

In allocation problem, the transformation function is, $(S_n - x_n)$

Now, one requirement for this multi-stage problem is that the state must be able to transform this state from one stage to another stage. What I mean by that is, let us say, you are in stage number n here, and this is stage number n minus 1. The state starting with a particular state S n, we must be able to get the state S n minus 1 for a decision that you take in this particular stage. This is the stage n and you are making a decision X n because of which the state gets transformed to S n 1. This is called as the stage transformation. So, you must be able to write the recursive relationship, you must be able to identify the state transformation.

What are the state transformation in water allocation problem? We had user 2 state number stage number 1, stage number 2 (()) finds your S 1. So, simply S n minus X n and in specific problems, the state transformation will be specific to that particular problem. For example, when we go to operation problem, this may be different. We look at the capacity expansion problem, this may be different and so on, but it involve that transformation of a state at a particular state, which defines the (()) for the next stage. So, this is called as the state transformation.

(Refer Slide Time: 44:45)



Dynamic Programming

- Recursive relationship identifies the optimal decision at stage n, for state S_n, given the optimal decisions for each state at stage $(n-1)$.

$$f_n(S_n) = \underset{\{x_n\}}{\text{Max}}\left[r_n(x_n) + f_{n-1}\{T(S_n, x_n)\}\right]$$

$r_n(x_n)$: Return for decision x_n in current state

$T(S_n, x_n')$: Transfer function to get the state S_{n-1} corresponding to S_n and x_n.

In allocation problem,

$$f_n(S_n) = \underset{\{x_n\}}{\text{Max}}\left[r_n(x_n) + f_{n-1}(S_n - x_n)\right]$$

So, you write the relationship state transformation and then, we have the concept of the recursive relationship. Now, the recursive relationship identifies the optimal decision at stage n f n of S n for state S n. So, this is the state S n in the stage n, given the optimal decisions at each state at stage n minus 1, there is the previous stage for which the calculations would have been already done. For example, in the water allocation problem we are proceeding in this direction. In stage 1, we had 1 user. We would have solved the problem and come to stage 2. When you come to stage 2, you have to relate the optimal decision which is for a given S 2, for the state S n, that is S 2 here, given the optimal decisions for each state at stage n minus 1.

So, stage 1, we would have solved for each possible state at that stage, and that is what we relate it with the optimal decision at stage n. Look at this expression now, f n of S n is the optimized objective function value at stage n and X n is the decision that we are making. So, when we write maximize over X n, this has to be read as maximize over X n and X n can be a vector. You may have a several decisions to make, but in the water user allocation problem, we had exactly one decision, namely the water allocated to the user at that stage n. So, f n of S n is equal to maximize over X n. That means you are making a decision on X n. Now, this term you understand correctly. R n of X n is the immediate return that you get at that stage because of the decisions that you make on X n. So, you make a decision of X n, you get a return of R n of X n at that stage. Associated with this decision now X n, you have to relate this f n to f n minus 1.

Now, this decision X n taken on the state S n transforms the state of the system to the previous stage, from this current stage to the previous stage and that transformation is given by t of S n, X n. Now this t of X n, S n is the state transformation equation. We will call it as state transformation. So, it is in fact, a function. It will be a function of S n, which is the state at that stage n and X n, which is the decision that you are making given the state S n. Now, the identification of the state transformation is an important concept here and that depends on the type of problem, actually the feature of the problem itself. Say for example, in the water allocation problem, we wrote this as the recursive relationship. We wrote as f n of S n at any given stage S, stage n here.

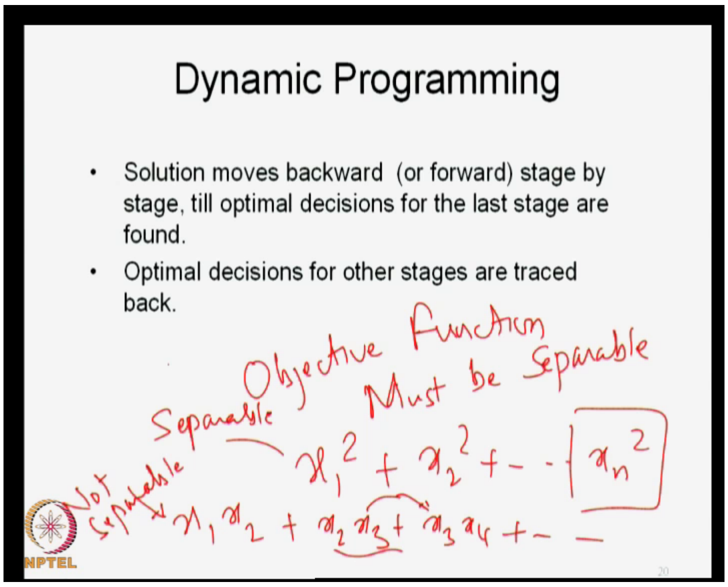
What are the state variable? A state variable was the amount of water available at that particular stage n. We wrote this as maximize over X n. Let us say, n was 2 second stage and X 2 which has the water allocated to user number 2 and you get a return associated with that decision X n plus f n minus 1, which is if we are looking at stage 2, f 1 n minus 1 will be f 1 of S n, which was the amount of water available to be allocated at stage number 2, which consisted of 2 users, namely user number 2 and user number 1. Because you allocated X 2 there, what is the amount that is left for user number 1, which is stage number 1? It will be S 2 minus X 2. So, that is an idea there and therefore, the state transformation for the water allocation problem is simply S n minus X n. So, this is how we define the state transformation for a given problem.

In the water allocation problem, I repeat the state transformation is simply S n minus X n. So, if you have several users, every time you relate this with the previous stage. Identification of the state transformation is an important exercise in writing the recursive relationship. Different types of problems will have different state transformations as we will see in the next problems that I will discuss. For example, the reservoir operation problem may have mass balance or continuity starting with a particular storage in a particular time period. You may want to relate it with the storage in the next time period, which in the backward direction will correspond to the previous stage and so on.

So, depending on the type of the problem, you must identify the state transformation. State transformation essentially indicates the transformation of the state starting with S n in stage n to state S n minus 1 in stage n minus 1 because we are relating it to stage n minus 1 in the backward direction. Once you identify the state transformation, it is easy for you to write the recursive relationship and writing the recursive relationship is the single most important feature of a dynamic programming, that is, identification of the recursive relationship is the single most important exercise that you do in dynamic programming.

Once you identify recursive relationship, the problem becomes very simple because in the discrete dynamic programming problem, state S n are discrete decisions, X n are discrete and therefore, once you write the recursive relationship correctly, all you have to do is solve this recursive relationship for various values of S n and each value of S n corresponding to a number of discrete values of X n. So, that is the concept of a state transformation and the recursive relationship. Now, there is a requirement here, which is often over looked and we tend to mechanically solve the problems. See what does this mean. This means that the objective function is in fact separable. That means, what we have been doing at stage n and what we have done at stage n minus 1 can be separated with one set of decisions taken at each of the stages, which means, which indicates that the objective function must be separable.

(Refer Slide Time: 52:35)



Dynamic Programming

- Solution moves backward (or forward) stage by stage, till optimal decisions for the last stage are found.
- Optimal decisions for other stages are traced back.

Objective Function Must be Separable

Separable $x_1^2 + x_2^2 + \cdots \boxed{x_n^2}$

Not Separable $x_1 x_2 + x_2 x_3 + x_3 x_4 + \cdots$

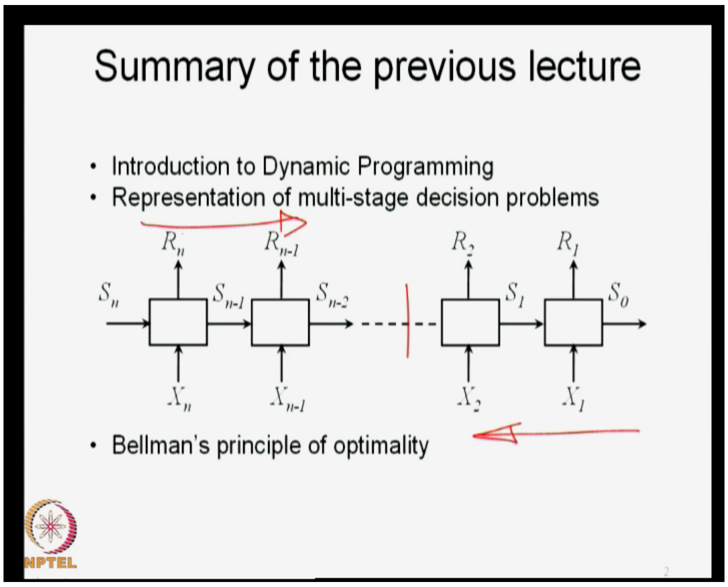
For example, look at this kind of function. I will say X 1 square plus X 2 square plus etcetera X 1 square. This I can break it into different stages. At each stage, I will consider only 1 of the variables. So, I will solve for X n, then I will solve for X n minus 1 etcetera. So, this is the separable objective function. However, I have some function like this. So, this is separable. Let us say, I have a function like this, X 1, X 2 plus X 2, X

3 plus X 3, X 4 and so on. If I have some such system, although I have different terms associated with different stages, I will still break it into different stages, but the objective function is not any stage. Let us say, this is dependent on the previous variable X 3, which appears in both the stages. Therefore, this is not sufferable and therefore, these problems are not to be formulated as dynamic programming problem.

So, one requirement is that the objective function must be separable and optimal decisions, so finally, what we do is, progress from one stage to another stage and then, complete all the calculations, complete all stages. Once you reach the final stage, you get the optimal objective function value and from that, you trace back to solution. So, the optimal decisions are traced back. Now, the solution moves backward or forward I say here. There are certain problems where you need to move only in the backward direction because the last stage, let us say, you are talking about time horizon. You may fix the boundary conditions to say, at the end of the last time period.

So, that time period boundary conditions may be known and then, starting with that, you may proceed backward or your initial storage condition. Initial storage condition in the case of a reservoir operation problem may be known, in which case you count from backward, in the backward direction and then, when you do the last stage calculation, you will use this condition that you satisfied. If on the other hand, you are looking at the shortest route problem let us say, then you may, it may be advantageous to proceed in the forward direction. So, you may either proceed in the backward direction or the forward direction.

(Refer Slide Time: 55:51)



Summary of the previous lecture

- Introduction to Dynamic Programming
- Representation of multi-stage decision problems

R_n
R_{n-1}
R_2
R_1
S_n
S_{n-1}
S_{n-2}
S_1
S_0
X_n
X_{n-1}
X_2
X_1
NPTEL

- Bellman's principle of optimality

However, your definitions of the state variables and the decision variables must be clear. What I mean by that is, that given a particular problem, you may either proceed in this direction that is you may specify X 1 various values and solve for this and then, go to this, the second stage for various values of S 2, you solve for these 2 together and so on or you may want to proceed in this direction, in which case you may start with S n minus 1 and related to S n. Similarly, S n minus 2 related to S n, S n minus 1 and so on. So, depending on the physical problem that you are solving, there may be advantage to either move in this direction or in the backward direction.

I will give another example of reservoir operation, where the direction of movement will become clearer. If we place one type of boundary condition so to say, that is at the end of the time period if I fix the storage, it may be advantageous to move in certain direction. If at the beginning of the time period you fix the condition, it may be advantageous to move in another direction and so on. So, much unlike the linear programming problems, there is some articulation that is involved in the dynamic programming formulations. It is not just an algorithmic way of solving as we did in the linear programming problem.

You must understand the physical problem correctly; see whether you can divide the problem into several numbers of stages, at each stage you are making exactly one policy decision and that is related with the previous stage computations and so on. So, you must make sure that the objective function is in fact separable and you are able to write the recursive relationship, you are able to write the state transformation equation from relating the transformation of the state from one stage to the next stage. In fact, the state transformation, the way you are able, you are writing the state transformation will also decide in which direction you would like to proceed in the computation.

So, the backward direction as well as the forward direction, you can move depending on the type of problem, and the Bellman's principle of optimality is valid in both the cases, but the way we state, may be slightly different. It is essentially the given the state of the system you must proceed in an optimal manner, whether you are proceeding in the backward direction or in the forward direction does not matter. All it says is do not worry about how you are arrived at particular state; you must proceed for starting with that state in that stage, you proceed in an optimal manner. So, this is the governing principle of dynamic programming.

In the next lecture, I will cover using the dynamic programming an important example for in water resources, that is, the reservoir operation problem. So, next lecture, we will start with the reservoir operation problem, we will solve with dynamic programming and then, see how we use this discrete dynamic programming problem for single reservoir operation. We will take it as an example and then in the subsequent lectures, we can talk about what we call as steady state operation and so on. So, this is the first example that I will be dealing with reservoir operation.

So, I will give some introduction to what we mean by reservoir operation, what are the decisions that we need to take, how we define the state variables and so on. So, essentially in today's lecture, we completed the water allocation problem that we took up in the last lecture, and we saw how we formulate the state transformation equation, how we formulate the, how we write the recursive relationship that relates computations from one stage to the next stage and so on.

Remember, in the type of problems that we are solving at any given stage, you just relate computations with what we have solved in the previous stage. So, just look up the previous table for the solution, pick up that particular objective function value and plug it in here. So, like this from one stage to the next stage to the stage, next stage, your computations proceed. However, you have to store all the stage computations until you complete all the stages. It is not that the moment I solve this stage computations, I will discard all the others, except I will return the previous one because only the previous one is not necessary.

Finally, you will have to trace back the solution and therefore, the computations that you have done right from stage number 1 to all the stages until this point stage number n have to be retained because finally, when you solve the last state computation, you have to trace back and the trace back involves tracing back solutions in all the stages. We will see what it implies in terms of computer memory, in terms of the computer time and so on. When I deal with other problems, we will discuss that feature. So, we will continue the discussion on dynamic programming in the next lecture. Thank you for your attention.