

Water Resources Systems
Modeling Techniques and Analysis
Prof P. P. Mujumdar
Department of Civil Engineering
Indian Institute of Science, Bangalore

Lecture No. # 16


Dynamic Programming: Reservoir Operation Problem

Good morning, and welcome to this the lecture number sixteen of the course, water resources systems modeling techniques and analysis. Now, we have been discussing over the two lectures - the dynamic programming, and in the last lecture we covered the water allocation problem using the dynamic programming completely. It was the example of allocating a known amount of water, among three users, and these three users has a side in the last class can be irrigation municipal, and industrial supplies, and the hydro power and so on. Towards the end of the last lecture, we just looked at the characteristic of the dynamic programming problems, much unlike the linear programming, dynamic programming requires some articulation to formulate the problems in the dynamic programming format. So, we will just again go through that quickly of **of** the characteristic of the dynamic programming problems.

(Refer Slide Time: 01:26)

Summary of the previous lecture

- Water allocation problem
- Characteristics of DP
 - A single n-variable problem is divided into n number of single variable problems.
 - Problem is divided into stages, with a policy decision required at each stage; objective function must be separable.
 - Each stage has a number of possible states associated with it.
 - Policy decision transforms the current state into a state associated with the next stage.
 - Recursive relationship identifies the optimal decision at stage n , for state S_n , given the optimal decisions for each state at stage $(n - 1)$.
 - 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 from the solutions of those stages.



2

So that the students will know given a problem, how we restructure the problem into a dynamic programming problem. Recall that a single n variable problem is divided into n number of single variable problems, in the dynamic programming. For example, in the water allocation problem what did we do, we had 3 decisions to be made x_1, x_2, x_3 corresponding to the water allocated to user 1, user 2, and user 3 respectively. So, this was the problem of 3 variables, what we did was we broke that problem into a single variable problem 3 number of them; at stage one - you solved only for x_1 , a stage 2 - you solved only for x_2 , stage 3 - you solved only for x_3 , at every time making sure that you will pick up the optimal allocations from the remaining stages. So, you are using the bellmen's principal of optimality to move in an optimal manner, starting with a given state of the system at that stage. So, in essence we have broken down a n variable problem into n number of single variable problems, that is what this state's.

Then the problem is divided into stages with a policy decision required at each stage. So, in the user water allocation problem, the water to be allocated to a user, become the policy decision. And the user sense us will define the stages. So, you had three stages in that particular problem; and these stages can be time units; for example, time periods, when we are talking about listener operation problems which are the introduced, the stages can be time or stages can be space. For example, you may look at water allocation among different crops. So, stages can be defined depending on the type of problem. Then each stage has a set of states associated with it. In the water allocation problem what were **what were** the states; the state variable at a stage was the amount of water available to be allocated to all the users, included in that stage.

This was the state variable. Now, the state variable can take on a the number of value, and in the particular class of dynamic programming problems, that we are talking about. Namely the discrete dynamic programming, we are dealing with only the discrete values of the state variables as well as the decision variables. So, water allocation problem the state variable could take values such as 0, 1, 2, 3, etcetera up to the maximum amount of water available we **(())** in that case 6 units. Then, we have the state transformation, starting with a given state you make a decision, because of which the state get transform; and these transformation is problem dependant. In the water allocation problem, the state transformation was simply $S_n - X_n$, where S_n was the state at the beginning of the time period - beginning of the stage, X_n was the decision of water allocation.

Therefore, the remaining amount of water that is left for allocation among all the other stages - the previous stage, it includes all the users is simply S_n minus X_n . So, the state transformation function or state transformation equation governs the transformation of the input state into an output state, because of the decision that you have made during that particular state. Then we should be able to write the recursive relationship; recursive relationship relates the computation of the objective function value for the current stage - **current stage** which is the objective optimal objective function value that is already obtained from the previous stage. So, this is the recursive relationship, then we can move the solution in the forward direction or in the backward direction or depend on as I mentioned in the last class, whether we move in the forward **forward** direction or in the backward direction will in general depend on the type of problem; and the type of boundary condition that we may have and so on. Boundary **boundary** conditions, I mean you may specify, let us say storage of the water available is known to be so much at the beginning of the time period or you may specify that at the end of the time period, I should have a minimum amount of so much amount of water left and so on.

So, you may apply conditions at the boundaries. So, depending on what type of boundary condition we may have the problem - for the given problem, we may it may be advantages for you to move in the backward direction or in the forward direction. Then once you solve for all the stages, as we explained in the last class; once you solve for solve the problem for all the stages, you trace that the optimal solution. So, essentially you have retaining the computations for each of the stages in tabular form, and till the problem is completely solved. So, let us say in the water allocation problem what did we do, we retain the computations or the complete table for the stage number one. We went to stage number two, picked up related with the solution obtained in the stage number one, solve for stage number two; went to stage number three, obtained the solution for stage number three using the solution that we obtained in the stage number two.

However, we retained stage number one tables, stage number two tables, stage number three table, etcetera until the complete problem is solved. You cannot discard the earlier tables, because we need that trace back. So, we need to trace back to optimal solution. In fact this is a correct occasion to tell that the dynamic programming, because of this requirement that you need to store the solutions, as you move from one stage to another

stage suffers from what is called as a curse of dimensionality. So, whenever you talk about dynamic programming, especially the discrete state dynamic programming's. It suffers from the curse of dimensionality, what we mean by that is, because of the requirement that you need to store all the solutions until, the problem is completely solved. As a number of state variables increased, the dimensions which you have store in the computer will keep on increasing. And especially in the discrete case, where corresponding to each of the state variables, you will have a number of discrete state - states possible for that particular state variable.

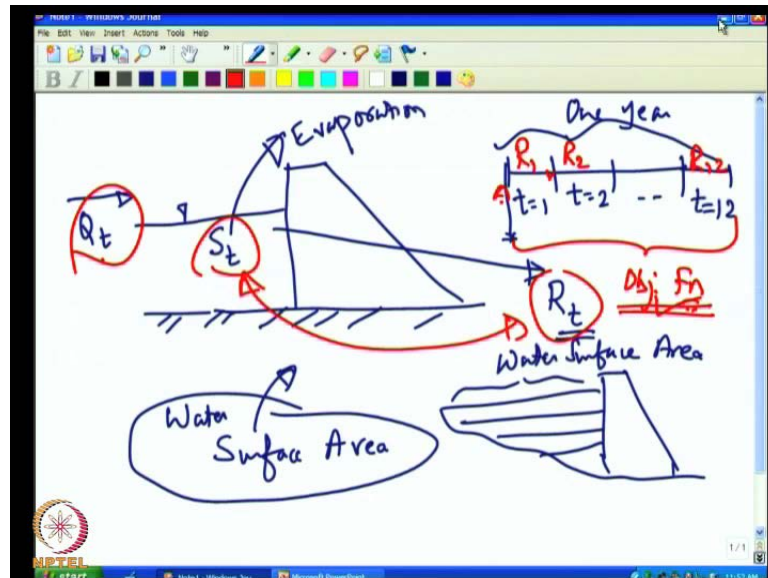
In the water allocation problem, we solve **solve** that the state variable can take on values 0, 1, 2, 3, 4, etcetera. The state variable in the water allocation problem was a single state variable - was single, namely the water available and for allocation. In many practical situations, you may need to account for several state variables; two state variables, three state variables, four state variables, and so on. As the number of state variables increased, and as the discrete cross intervals or the discrete states that particular state variables can take on increase, as **as** the number increases. The dimensions that you have to store or **or** the storage dimension on the computer, will keep on increasing exponentially.

And this is what is called as curse of dimensionality, because of which the number of state variables is that we often use in the dynamic programming is limited to about 2 or 3 and so on, although in some applications, we have use in our institute to have to used as many as four different state variables with large number of discrete states. I will perhaps discuss this type of application later on, and see the implications of the computer memory and so on. Of course, with the available computer resource is, it is not extremely difficult this days to **to** solve for dynamic, solve the dynamic programming problems; even with a number of stage - number of state variables.

However, you must remember dynamic programming's suffers from curse of dimensionality; with that background now, we will go to another application of the dynamic programming. An important example in the dynamic programming that of reservoir operation, because I am introducing the reservoir operation the first time, with an example of application of dynamic programming. Let us first get a broad background on what is the reservoir operation problem. In the subsequent lectures, many times we will review that this problem; that is many applications of the techniques, we will

applying to reservoir operation problem. Many applications are dealing with reservoir operation problem, and let us see what is the reservoir operation problem?

(Refer Slide Time: 11:29)



So, let us look at some get some preliminary background on what is the reservoir operation problem? You have a reservoir by reservoir, I mean you have a dam, and be behind the dam you are storing the water. Now, most of the surface water systems that we deal with will have a surface reservoir, which stores a large amount of water. In fact, water resource systems - especially the surface water system almost in variably involve reservoirs which have huge amount of water stored. And the planning for the operation of such reservoir is an important, and absolutely not trivial problem. Because you are dealing with large amount of water, and the allocation of this amount of water for different uses, during different time periods, given that the storage is continuously fluctuating depending on the amount of inflow, that is coming in the amount losses that are taking place, and so on.

This problem is a non trivial problem, deciding on the releases during time period. Let us say that S_t is the storage at the time period t , at the beginning of time period t , and these from these S_t which also has a surface area, that is the water surface area. As you know a formula we have a initial water recourse engineering or hydrology course, you would have seen this, the typically the contours will be something like this just, behind that the lower. So, at different levels of storages - these are different levels of storages, you will

have a different water surface area, this is the water surface area. Now, depending on the water surface area, you will have losses taking place vapor transportation, evaporation I am **sorry**, evaporation losses will be taken place.

And that will be governed essentially by how much is the water spread - **water spread** area or water surface area. How much is the water spread area that will decide on the evaporation, and of course **of course**, evaporation also depends on the time period t or different time periods the rate of evaporation will be different. In addition there is the inflow that is coming in here. So, will denote the inflow as t . So, we are talking about different time periods t is equal to 1, t is equal to 2, etcetera. And if we are talking about a monthly operation in 1 year, this may be 1 year for a sense. You will have Q_1, Q_2, Q_3 , etcetera, Q_{12} during different time periods.

So, starting with a particular storage, let us say that you know the storage at the beginning of the time period. You need to make decision on how much to be released from the reservoir, how much amount of water to be released from the reservoir. During time period t . So, we have to make a decision on R_1, R_2 , etcetera, R_{12} ; set that the total objective function value for this entire arisen time arisen, may be 1 year, may be 10 years, may be 12 years, and so on is maximized. So, we are looking at an objective function value, and we would like to maximize this objective function value.

If we are talking about benefits arising out of the release of t ; for example, you may be releasing the all t here for hydro power generation, in which case you may **you may** have your objective function as maximize the amount of hydro power generation. On the other hand, you may be releasing R_t for irrigation purposes, in which case objective function can be related to the crop is that you may get, and so on. So, the objective function value when we are maximizing, may be in general related to the benefits either tangible or insensible intangible occurring out of the release that you have made R_t , and the storage that you have S_t .

In general, the objective function value will depend on S_t as well as R_t . Now, look at what is happening here, from one time period to another time period, as you move the storage keeps changing. Let us say you had a given storage S_1 at this location, because you got a inflow of Q_1 , and you made a release of R_1 for that time being ignored in the evaporation, and other losses. The storage at this point gets fix. So, the storage at this

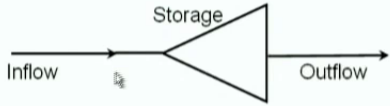
point will be a function of will be determined in fact, they **they** storm what was the initial storage - what was the inflow, and what is the release that you. To release that you made from the there are simple mass balance, starting with given storage you add the inflow take out. If you are accountant for the losses, take out the losses, and take out the release. So, that will decide the storage at the beginning of the next time period. So, like this from time period to time period to time period the storage keeps on changing, and you need to make the decision on how much to be released, during each of the time periods. They storm the storage that exist during that the beginning of the time period, and the inflow that is likely to come during that particular time period. So, this becomes a multiple stage problem - multi stage decision problem, as you can see you start with a given storage based on that storage you make a decision account for the inflow, that is coming and **(())** accounting for losses and so on.

Because the free stress states get transforms to the another storage at the end of the time period, which is also the beginning of the next time period. Based on these transformed state, you make another decision R 2 that say you started with R 1, you make a decision R 2, change the transform to S 3, make a decision R 3 and so on. Like this from 1 state 1 stage to another stage to another stage based on the state of the system at the beginning of that stage, you make decision such that when you reset end of the arisen. In this particular case, one year arisen when you reach the end of the arisen, your objective function value for the entire problem is maximized. So, these kinds of problems are eminently **emanable** to be solution to be solved by dynamic programming problems. So, let us so look at, how we address this problem using the dynamic programming that you have introduced in the previous lectures.


(Refer Slide Time: 19:16)

Dynamic Programming

Reservoir operation problem:



- Operating policy: Amount of release from the reservoir during a period (e.g., a month, a season etc.), for a given storage level at the beginning of that period.
- Stage: Time period (e.g., month) for which decisions are required.
- State variable: Storage at the beginning of a stage.
- Decision variable: Release from the reservoir during a period.



3

So, as I said the reservoir operation problem is, that you have a storage here, you have an outflow; this may include the you may have a release plus spill or overflows. What you mean by that is, that you may have a controlled release, which means you made operate the guides, and then make that to release. And you may also have spill; that means, the storage has been exceeded, and still the flow is coming and therefore, you allow it to spill lower. So, the outflow here will consists of release as well as spill or the overflow. Now, we define the operating policy for the reservoir, as the amount of release from the reservoir during the time period. For example, you may want to operate this reservoir on a ten daily bases for irrigation purposes, let us say that every ten day period I will release some amount of water uniformly.

That means, it is on that I will wait for the ten days, and then in 1 pulse I release the water know. You decide on the amount of water to be released during the ten days, and release that amount uniformly, such that the over ten days total amount of release is equal to what we had this days. So, the release policy is the amount of release to be made from the reservoir, during certain time periods. And these time periods can be months can be ten days, and some times in the **in the** planning situation, you may talk about seasonal releases and so on. And sometimes, in the flood control context you may talk about hourly release or daily release and so on. So, depending on the tying interval of interest, your stages in the dynamic programming gets fix.

The stage then that is in the dynamic programming as I mentioned in the last class, you have to decide on when **when** you are looking at dynamic programming problem, you must be clear on what is the stage of the dynamic programming - what is the state variable - what is the decision variable, and what is the recursive relationship? What is the state transformation, and so on. So, these are the important characteristic of the dynamic programming, that you must be able to write down as a problem formulation. So, the stage in this reservoir operation problem is the time period during which the releases are required. Let us say here talking about monthly time periods; what is the monthly release that has to be maintained? June month, July month, August month and so on, so that becomes the stage.

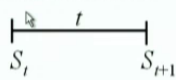
The state variable to start with we will start with simply the storage - **storage** at the beginning of a stage or storage at the beginning of a time period, that defines state variable. As the problem becomes more and more sophisticated or more and more complex, you may want to add different state variables. However, we will start with only one variable now, which is the storage at the beginning of the time period. Which means what? The inflow during the time period is known, and therefore the total amount of water available, during the time **period** at the beginning of the time period will be known. So, in this simple example in this introductory example on reservoir operation, we will assume that the state variable is the storage at the beginning of the time period, which is also the stage. That means, storage at the beginning of the stage. And the decision variable is the release that has to be made from the reservoir, during the particular time period.

The inflow and the release happen continuously, so for the inflow as well as the release we talk about release during the time period, inflow during the time period, and so on; where as the storage is the **(())**. Therefore, we talk about storage at the beginning of the time period, storage at the end of the time period and so on. Although there is also a continuous variation of storage, because we are making release continuously, there is an inflow that is continuously occurring etcetera, but we recon the storage is at the end of the stages or at the beginning of the stages **alright**.

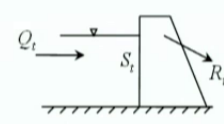
(Refer Slide Time: 24:00)

Dynamic Programming


State transformation:



Storage continuity (Mass balance)


$$S_{t+1} = S_t + Q_t - R_t \quad \dots\dots \text{Neglecting losses}$$

where S_t : Storage at the beginning of period t
 Q_t : Inflow during period t
 R_t : Release during period t



4

Let us see, what is the state transformation for this example? As I said starting with S_t which is the storage, you make release R_t , and also there is a inflow that is coming Q_t during the time period t . So, this is the time period t , you started with the storage S_t , and end up with the storage S_{t+1} . So, this state S_t at the beginning of the time period t gets transform to the state S_{t+1} , and the stage transformation is simply govern by the mass balance. So, we can write S_{t+1} is equal to S_t which has the beginning of the time period storage plus Q_t which came as inflow during the time period which contributed to the storage minus R_t , which you have to taken it out from the storage neglecting losses. So, if I had losses you would have written minus Q_t and so on, for some purpose we neglect losses right now. We will of course, introduce how to accounts for losses, and so on in subsequent classes. Right now, the focus is on understanding how use dynamic programming for this simply stated problem.

So, storage remember storage is at the beginning of time period. So, at the beginning of time period you started with S_t , you added Q_t , and took out R_t . So, that will define the storage at the end of the time period t , which is also the storage at the beginning of the next time period S_{t+1} . Therefore, starting with the state S_t , you have obtained the state S_{t+1} , and that is what is called as the state transformation. So, you know the state transformation equation are there is the add operation problem.

(Refer Slide Time: 25:54)

Dynamic Programming

Recursive relationship:

$$f_t^n(S_t) = \underset{\substack{0 \leq R_t \leq (S_t + Q_t) \\ S_t + Q_t - R_t \leq S_{max}}}{\text{Max}} [B_t(R_t) + f_{t+1}^{n-1}(S_t + Q_t - R_t)]$$

Total water available in period t

$B_t(R_t)$: Benefits associated with release R_t in period t

$0 \leq S_t \leq S_{max} \quad \forall t$

S_{max} : Reservoir capacity

$\{Q_t\}$ known

NPTEL

Now, we have several states like this, the state t is equal to 4, 3, 2, 1 etcetera; 1, 2, 3, 4 it is progressive in this direction. The last time period in general you may denote it as t is equal to t . Let say keeps on going, and n is equal to one is that, therefore we will write t is equal to t ; this is the time period corresponding to last stage or the last time period I will say... For example, in a monthly case this should be 12, t is equal to 12. So, like this, we associate stages n is equal to 1 where progressing in the backward direction; n is equal to 1, n is equal to 2, n is equal to 3, etcetera. Whereas t is equal to 1, t is equal to 2, this moves in forward direction. That is the time progresses in forward direction, computation progress in the backward direction.

You may have a beginning of the period storage no more, that is at the beginning of the arisen - time arisen that we are looking at, this storage is known. And you are progressing from this direction. So, at any given stage, you can write the recursive relationship, n denotes the stage, and t denotes the time period. So, we will write this as f_t^n - this is a subscript, this is the superscript. So, we are saying that t and n are related, as shown here. For example, t is equal to 4 corresponds to n is equal to 1, t is equal to 3 corresponds to n is equal to 2, and so on. What is it that we are looking at our decision is R_t , because you make the decision R_t in a particular stage n , which corresponds to the time period t .

You get a benefit from that particular decision d_t of R_t plus you would have solve for solved in the previous state. Let us say, that you were looking at this particular state. You have progressing the backward direction, therefore you would have solve for previous stage; the previous stage would have included everything until the end of the time arisen until all the stages are completed. So, in the previous stage, what happens? The time period is $t + 1$. So, this is time period t , the previous stage which is n is equal to 2 let us say, from n is equal to 3, n is equal to 2 we are relating. The time period will be $t + 1$, whereas the stage will be $n - 1$. Now, we use both these indicators n as well as time, just make sure that you are keeping track of the position of the computations.

This should be S_{t+1} , because we are talking about $t + 1$, and you are objective function value is related with S_t which is the state variables. So, on the state variables S_t , we are making the decision R_t , and relating it with the previous stage computations which correspond to time period to $t + 1$. And they the storage at that time period $t + 1$ is S_{t+1} , which is governed by the state transformation $S_{t+1} = S_t + Q_t - R_t$. So, this in fact, gives you S_{t+1} . Now, like we did in the water allocation problem earlier. we solve this for various possible values of the state variables at a particular stage, which means we will start with different values of S_t .

Now, what are the bounds on the S_t , you have the S_t which is the storage at the beginning of the time period, and we are talking about live storage only will not worry about the dead storage. That means, live storage is the storage that is actually available for consum (C) may be irrigation, hydropower, and so on. So, the actual available storage is what we are talking about not that dead storage. So, that storage can vary from 0, 2; let us say this is the capacity, we write it as S_{max} . So, the storage can take on any value between 0, and S_{max} . That is what we write here, $S_{t+1} = S_t + Q_t - R_t$. That means, starting with the given storage, you added the flow Q_t , and you took out to release R_t , which also includes the overflows.

That will define the end of the period storage that should be less than S_{max} , that is what we are saying. So, this is what we write, you are searching for all such variables such value of S_t , which will satisfy this condition $S_{t+1} = S_t + Q_t - R_t$ which is the decision that what we are making must be less than or equal to S_{max} ; S_{max} is the maximum storage. Then we are making decision on R_t , relate this with what we have done earlier in the water allocation problem, now we are solving this problem for various

values of S_t which will satisfy this condition. Q_t is known, and R_t is the decision that we are making. So, we have to satisfy this condition in choosing the value of S_t - discrete value of S_t . What are all the values at R_t can take, R_t is the release that is the main during the time period t .

What are all the values that R_t can take, if you look out this figure now; the water that is available is S_t plus Q_t , during the time period the flow has come and added to the storage, because the S_t earlier. So, S_t plus Q_t is the total amount of water that is available for allocation or for release. So, your R_t value which is the decision variable can take an values between 0, and S_t plus Q_t . So, this is the S_t plus Q_t , and that is the total water available in period t in storage which accounting for inflow. So, R_t ; that means, we can release only up to S_t plus Q_t during the time period, that is what this stage. And S_t such that, it governs the mass balance subject to the condition that your storage does not exceed the maximum storage, and that this condition.

So, this is the recursive relationship that you are write; you are relating what is acting at time period t to what has happen during the time period $t + 1$ in the backward direction. That means, from stage n calculation, we are relating it with the previous stage calculation $n - 1$ at stage calculation, in this the B_t , R_t are the benefits associated with release R_t . In this simplistic problem **problem** that I am talking about. In fact, they may be associated with storage as S_t alpha as I said, let say that we are looking at hydropower, releases for the hydropower. Now the hydropower releases will also depends on the head, and therefore on the storage. So, in general it may the benefits are the returns that are the acquiring out of such a problem, the out of such of decision that you are making out namely R_t will be dependent both on the state variables as well as on the decision variables.

But let as **right**, let as start with simple problem in which the benefits are the returns that are you acquiring are depended only on R_t , which is the decision that we will make. Then you have this condition $0 \leq S_t \leq S_{max}$. So, you will choose your S_t values, such that at no time you are exciding the maximum storage - maximum live storage. And we are dealing with only the live storage that is why you will right $0 \leq S_t \leq S_{max}$ the. Q_t sequence here, which is the inflow during the time period, Q_t is the inflow during the time period - time period t , this sequence is known.

Now, in actual problems there are lots of not necessary complexities, uncertain unassociated with this Q_t . We will deal with those type of problems, when I come to stochastic dynamic programming and so on, in the **in the** same course, but right now we assume that it is the deterministic problem, in the sense that Q_t values all are known, the sequence Q_t is known. And S_{max} is the reservoir capacity that is known. So, in this recursive relationship then for a given S_t , we are making decision on R_t ; such that the total objective function value until the end of the Harrison is maximized. And we are using the Bellman principle of optimality, the immediate benefit said a Q out of making the decision R_t plus the optimized benefits that you have already obtained, during the previous stage computations which correspond in this particular case to the next time period.

So, this is how we apply the **the** this is how we use the Bellman principle of optimality in reservoir operation problem. What we do in this case is, we are progressing from the last stage that is the last time period which corresponds to stage number one. So, this stage one, stage two, etcetera, are the stage in the dynamic programming competition. So, n is equal 1 correspond to the last time period t , t is called to 4 this case. We solve this problem looking at only the last time period, where progressing in this direction now. We look only at the last time period. So, at n is equal to 1, we solve this for different values of S_4 ; we say if S_4 equal to 0, what should be my R_4 ? S_4 is 1, what should may for S_4 and so on.

For different discrete value of S_4 , you obtain the decision R_4 here. Such that, because we are starting with the first period, you will have only this term; this term is not existence, because there is **there is** last time period. Then having solve this, you come to this is stage now, that is n is equal to 2, and the time period is 3. So, you will stand here, in which case you will include both this together, and so on. So come to this include all these three together, and so on. So, this is how much the same way as we did in the user water allocation problem, this is how we progress in the backward direction.

(Refer Slide Time: 38:02)

Dynamic Programming

Example:

- Inflows during four seasons to a reservoir with storage capacity of 4 units are 2, 1, 3 and 2 units respectively.
- Overflows from the reservoir are also included in the release.
- Reservoir storage at the beginning of the year is 0 units.

Q_t

$S_1 = 0$

$n=4$ $n=1$

$t=1$ $t=2$ $t=3$ $t=4$

2 1 3 2

4

4

NPTEL

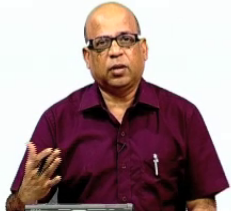
Let us take an example which will make this competition clear. We will take 4 time periods; t is equal to 1, 2, 3, 4, etcetera. So, we are saying t is equal to 1, t is equal to 2, etcetera, t is equal to 4. And n is equal to 1, n is equal to 2, and so on, n is equal to 4. So, there are 4 seasons to a reservoir, and the inflows are known. So, inflows are here Q_t , I write this as Q_t here, this is 2, 1, 3, 2. So, the inflow during in the time period 1 is 2, inflow during the time period 2 is 1, and so on. So, the reservoir inflow is known, the storage capacity is known. We are talking about a maximum storage of 4 units, and out flows from the reservoir are also include in the release; that means, if you have a excess of let says that you were already at 4, and 2 units of flow comes. Then the whole two goes as out flow, if you want to returns the storage at 4 itself. The reservoir storage at the beginning of the here is 0 units; that means, at this points we are saying that S_1 is equal to 0. So, this is the given condition - given boundary condition. So, this is the problem now, and what is it that you want to decide, you have to decide on the release sequences, what should be may R_1 , what should be may R_2 etcetera; such that the total benefit is maximized.

(Refer Slide Time: 39:57)

Dynamic Programming

- Release from the reservoir during the season results in the following benefits which are same for all the four seasons.

Release	Benefits
0	-100
1	250
2	320
3	480
4	520
5	520
6	410
7	120



And the benefit function is given here which for this particular simple example, we will assume that it is the same for all the time periods. In general in practical applications, you will have different benefits associated with different time periods. But for this example, we will assume that the benefits remain the same for a **for a** given amount of water that you use release from the reservoir. So, what does this table give, these says that if you release an amount of 0 - the benefit is minus 100, if you release an amount of 1 - the benefit is 252; it is 323, it is 480 and so on. So, for discrete values of the releases, you have the associated benefits. These are all in the command rate units, these may be (()) units, and these may be monetary units.

Now, why did we go up to 7, you just look at these do not lose height of the physical problem; the physical problem is diet, there is a story, there is continuous of fluctuations of the storage taking place, continuous change of storage is taking place, because of the inflow, and because of the release. The maximum storage is 4 units, the inflow during the 4 seasons are known, and those inflows are 2, 1, 3, 2. So, what is the maximum inflow - maximum inflow is 3. And therefore, the total amount of water that you can release, the maximum amount of water that you can release including the overflows, at any given time period is 4 which is the maximum storage plus 3 if it is the maximum inflow.

And therefore, the benefits have to be desired given for 4 plus 3 – 7, 7 units, and that is why we have the benefit function for 7 units. So, these are the benefits associated with release, and as I said release also includes the compulsory overflows which are over and above the maximum storage.

(Refer Slide Time: 41:52)

Dynamic Programming

- To obtain the release policy backward recursive equation is used, starting with the last stage.

$$S_1 = 0 \quad \left| \begin{array}{c} t=1 \\ n=4 \end{array} \right| \begin{array}{c} t=2 \\ n=3 \end{array} \left| \begin{array}{c} t=3 \\ n=2 \end{array} \right| \begin{array}{c} t=4 \\ n=1 \end{array} \right|$$

$S_2 \quad S_3 \quad S_4$ ← Progress of computations

Stage 1:


$Q_4 = 2 \quad t = 4 \quad \text{and} \quad n = 1$

$f_4^1(S_4) = \text{Max} [B_4(R_4)]$

$0 \leq R_4 \leq (S_4 + Q_4)$

$S_4 + Q_4 - R_4 \leq 4$

Total water available for release.
Max. storage.



So, we start with this now, and proceed in the backward direction. So, in the last stage, the last time period which corresponds to n to is equal to 1, we are solving the problem for this time period only. The flow is known in this time period, which is 2 units - Q 4 is 2 units, t is equal to 4 which is the time period, and n is equal to 1 which is the stage in the dynamic programming. So, at this is stage, we solve the dynamic programming we write in our general notation f 4 1, which means this is the time period 4, and this is the dynamic programming stage 1 for a given S 4, we are solving this problem for a given S 4 is equal to maximize B 4, R 4, because there is no other time period left now.

This is the last time period stage number 1, therefore we write this as B 4, R 4, B 4 of R 4, where B 4 of R 4 is the benefit associated with release of R 4 in time period 4. And R 4 has to satisfy the condition, that it should be non-negative, and it should be less than or equal to S 4 which was the storage plus Q 4 which is the inflows. So, together this is the water available - total water available for release. And the S 4 that you choose should be such that S 4 plus Q 4 minus R 4 should be less than or equal to 4, which is the capacity of the reservoir, this is the maximum storage. So, you choose those values of S 4, which

satisfy **satisfy** these conditions, and among the S_4 you make decision on R_4 ; various possible values of R_4 you choose, and then pick up that particular value of R_4 for a given value of S_4 , which will maximize B_4 of path. So, once you understand this path correctly, the solution in other mechanical much the same way as we did for the water allocation problem. So, what is it that we are doing? we are talking about time period 4 now, stage number 1, the Q_4 is equal to 2 that is the inflow during the time **time** period is 2.

(Refer Slide Time: 44:48)

$Q_4 = 2$ **Dynamic Programming**

S_4	R_4	$B_4(R_4)$	$f_4^*(S_4) = \text{Max}[B_4(R_4)]$	R_4^*
0	0	-100	320	2
	1	250		
	2	320		
1	0	-100	480	3
	1	250		
	2	320		
	3	480		
2	0	-100	520	4
	1	250		
	2	320		
	3	480		
	4	520		

Contd. 9

We will solve this for various possible values of S_4 ; that means, what we are saying is if my S_4 which is the storage at the beginning of the time period 4 is 0, how much should I release. If my S_4 is 1, how much should I release, and so on. It can go up to 4, because the storage values can be only go up to 4 – 0, 1, 2, 3, 4. So, we will solve this for different values of storage, possible storage values at the beginning of the time period 4. Look at this, if my S_4 is 0 which means that the storage at the beginning of the time period 4 is 0, we have an inflow 2, and therefore the release value R_4 can take on values up to 2; namely 0 plus two. So, it can be either 0 or 1 or 2. So, what we are saying is if my S_4 is 0, R_4 can be either 0 or 1 or 2. If R_4 is 0 my benefit is minus 100, if R_4 is 1 benefit is 250, if R_4 is 2 benefit is 320.

And therefore, the maximum value corresponding to this is 320, and the R_4^* star here corresponds to the R_4 that results in this maximum value. So, that is how we get R_4^* star


or relate this again with the water allocation problem that we are solved earlier. Then S_4 is equal to 1, 1 plus 2 that is 3; therefore, my R_4 can be 0, 1, 2, 3, for you can release up to 3. And these are the associated benefits, and this is the maximum value, and this is R_4^* should be 3. So, if the storage is 1, then my release should be 3; is what this column size, this is a row size. S_4 is equal to 2, then again my release can be up to 2 plus 2 which is 4. So, 0, 1, 2, 3, 4, and these are the associated benefits **benefits**, this is the maximum benefits, and this is the release. Which means that, it says if you are storage is 2, release 4, now we will go to storage is equal to 3. Remember here all of these we started with a release of 0; that means, 0, 1, 2; 0, 1, 2, 3; 0, 1, 2, 3, 4, etcetera.

(Refer Slide Time: 47:32)

Dynamic Programming

$Q_4 = 2$
Contd.

S_4	R_4	$B_4(R_4)$	$f_4^*(S_4) = \text{Max}[B_4(R_4)]$	R_4^*
3	1	250	520	4,5
	2	320		
	3	480		
	4	520		
	5	520		
4	2	320	520	4,5
	3	480		
	4	520		
	5	520		
	6	410		


10

When we come to 3; **3** plus 2 is 5 which is more than the maximum storage. And therefore, if you release 0 there, what happens? Your end of the period storage will be more than the capacity which is not acceptable. And therefore, we start the release with one, essentially we are satisfying this condition now; R_4 should be less than or equal to S_4 plus, that is S_4 plus Q_4 minus R_4 should be less than or equal to 4. This is the condition that we are satisfying and therefore, R_4 cannot be 0; if R_4 is 0, then this condition will be violated. And therefore, we for a S_4 is equal to 3, we start with R_4 is equal to 1, you can have 1, 2, 3, 4, 5; that is 3 plus 2 is 5, you can go up to 5.

Similarly, when we come to S_4 is equal to 4, and these are **(())**. So, you get a maximum benefit of 520, and 520 corresponds to 4 or 5 which means that it says, if my S_4 is equal

to 4 either release 4 or 5. And you get a maximum benefit of 520, when we come to 4 here, what is the total water available 4 plus 2 there is 6. And therefore, you cannot release 0, which will lead to the end of the period storage as 6, you cannot release 1 which will lead to the end of the period storage as 5 which is again more than 4, 4 which is the capacity. And therefore, you start with the minimum release up to. So, 4 plus 2 minus 2 that will be 4 only. So, that is acceptable. So, go 2, 3, 4, 5. Why do we go up to 6, because the total amount of the water available is 6 - 4 plus 2 that is 6.

And these are the associated benefits, and this is the maximum benefit and corresponds to these corresponds to 4 and 5. So, the first stage computation are fairly straightforward; we simply look at all the possible values of the releases, and pick up the associated benefits, pick that particular value of R 4 which gives you the maximum benefits associated with this. So, that is fairly straightforward, because you did not have any other time period beyond this. So, you are just looking at one time period, when we go to the next time period which is the next stage here, in this case n is equal to 2, which corresponds to t is equal to 3, here at this point. So, you will solve for various values of S 3, and include both these time periods together. So, that is what we do for the next time period.

(Refer Slide Time: 50:20)

Dynamic Programming


Stage 2:

$$Q_3 = 3$$

$$t = 3 \text{ and } n = 2$$

$$f_3^2(S_3) = \text{Max} [B_3(R_3) + f_4^1(S_3 + Q_3 - R_3)]$$

$$0 \leq R_3 \leq (S_3 + Q_3)$$

$$S_3 + Q_3 - R_3 \leq 4$$

11

So, stage 2 that is n is equal to 2, t is equal to 3, time period is 3, where inflow is 3 units; you are writing for S 3 for a given value of S 3, this is the time period, and this is the

stage. We are writing the recursive relationship for this particular stage now, f_3^2 of S_3 ; that means, if S_3 is known, then you are looking at a decision R_3 which will maximize B_3, R_3 plus as a result of making this decision, you would have ended up in a storage S_4 , which is S_3 which is known, plus Q_3 which is known, minus R_3 which is the decision that you are making now.

Therefore, that value is known, this will define S_4 and for S_4 you have already solved, and that is f_4^1, f_4 and that is f_4^1 of the end of the period storage. So, at this stage what we are looking at is maximum, you are **you are** making a decision R_3 corresponding to a given S_3 , which will maximize B_3 of R_3 plus the optimized value of the objective function for the previous stage one. So, from stage 2 you are relating it to stage 1, which is from time period 3, you are relating it to time period 4. And again all these similar constraints; that means, you are S_3 that you are choosing here, must be such that S_3 plus Q_3 which is known minus R_3 which is the particular value of the R_3 that you are looking that, must be less than or equal to 4.

So, for a given S_3 like we did in the previous table, you choose R_3 such that this condition is satisfied, Q_3 remaining constant. And R_3 the values of the R_3 which is the release during the time periods 3, must be such that you can go from 0 to S_3 plus Q_3 which is the total amount of water available. Once you understand the computation for the stage two, you can carry out computational for any number of stages. So, we are essentially now talking about time period 3, and n is equal to 2, Q_3 is known; Q_3 is 3.

(Refer Slide Time: 53:03)

Dynamic Programming

$Q_3 = 3$ $f_3^2(S_3) = \text{Max}[B_3(R_3) + f_4^1(S_3 + Q_3 - R_3)]$
 $0 \leq R_3 \leq (S_3 + Q_3); S_3 + Q_3 - R_3 \leq 4$

S_3	R_3	$B_3(R_3)$	$S_3 + Q_3 - R_3$	$f_4^1(S_3 + Q_3 - R_3)$	$B_3(R_3) + f_4^1(S_3 + Q_3 - R_3)$	$f_3^2(S_3)$	R_3^*
0	0	-100	3	520	420	800	2, 3
	1	250	2	520	770		
	2	320	1	480	800		
	3	480	0	320	800		
1	0	-100	4	520	420	960	3
	1	250	3	520	770		
	2	320	2	520	840		
	3	480	1	480	960		
	4	520	0	320	840		

Contd.

And this is the regressive relationship that we are talking about. We solve this problem for various specified values of S_3 , which means essentially what we are saying is that if, the storage at the beginning of the time period 3 is known to be 0 units, 1 units, 2 units, 3 units, and 4 units; there is a maximum storage that is possible. How much should I released during the that time period R_3^* , such that the immediate benefit that you get out of that particular decision R_3 , and the optimized decision that you have you would get from the previous stage. Corresponding to the storage that results out of S_3 , and R_3 together for this Q_1, Q_2 ; that is what that is the question that you are asking.

So, how do (()) I do, in these are typically solved using the tabular form. So, you just look at these and set up your table. So, you are solving for a given S_3 value. So, you put a column for S_3 , you are making a decision on R_3 put a column for R_3 ; for a given R_3 , you know what is the benefit B_3, R_3 . This is given by this table - initial table, this remains constant for all the time periods. So, you pick up that, then from S_3 for a given Q_3 , you have made a decision R_3 therefore, you end up in storage S_3 plus Q_3 minus R_3 at the end of the time period. You set up a column for that, corresponding to this storage, you have to pick up the objective function value from the previous stage n. So, pick up set up column for that S_4 1 of S_4 plus Q_3 minus R_3 .

And then you add these 2 B_3 plus these. So, you add these 2 terms set up column for that; pick up the maximum associated with this. So, set up column for that, and then this

is the R_3 star that results in this particular value. So, this is what you do for various given values of S_3 . So, we start with 0 plus 3. So, you can your release can be up to 3 – 0, 1, 2, 3, and so on. So, we will start with the stage 2 computations, because stage two computations are important for understanding the flow of the dynamic programming computations.

So, in the next lecture, we will begin with the stage two computations. So, essentially what we did in our today lecture is that we introduce the reservoir operation problem, remember the reservoir operation problem is classical water resources systems problem. And we are talking about a single reservoir problem in this case, and very simplified in the reservoir problem, where multi stage decision need to be taken. Typically we are taking decision across time periods. So, a time period constitutes a stage in the dynamic programming, the state of the system is defined by the storage at the beginning of the time period. The decision that were making is the release during the time period, and in this particular types of problems that I am introducing introduced in this particular lecture, we are ignoring the evaporation.

Are you there are very very elegant ways of accounting for the evaporation losses, as storage dependant losses which will cover later on in when we deal with application; deal with real time, real life applications. Then we set up this problem as dynamic programming problem, as a multi stage decision problem; start with the stage number 1 which corresponds to the last time period t , you proceed in the backward direction. Your storage at the beginning of the year, which is the storage at the beginning of time period t is equal to 1, which corresponds to the last stage in the dynamic programming is known.

You may have several different types of boundary conditions, which are discussed after the problem is completed. So, starting with the last stage, last time period which is the stage number 1, what we do is that you simply look at all the possible values of S_4 in that case, which has the time period 4. And look at all the fusible values of R_4 , and pick up corresponding to each of this S_4 for a given Q_4 , Q_4 is known in this particular case, for a given Q_4 you pick up all the possible values of R_4 , and pick up that particular value value of R_4 as the optimal release which corresponds to the maximum benefit, because we do not have anything look beyond, this is the last period. And then, we need to connect this computation with the next stage computation, which corresponds to the previous time period. This we will do in the next class. So, the water allocation problem,

and the reservoir operation problem are classical problems in water resource system; and the both these are amenable for a legend solution - a legend formulation using the dynamic programming.

And I also mentioned about the curse of dimensionality, and dynamic programming problems, which often limits its use in water resource system, because in water resource system for realistic formulations of the problem. You need to account for not one state variables, but several state variables; for example, storage at the beginning of certain time period can be a state variable inflow, can be a state variables rainfall in the command area, can be state variables soil moisture, can be a state variables and so on. Because you want to make the optimal decision based on all these values, and as the number of state variables increases, the increases the dimensionality of the problem increases. And therefore, it leads to what is called as curse of dimensionality. So, we will continue this discussion in the next class, **thank you** for your attention