Dynamic Programming

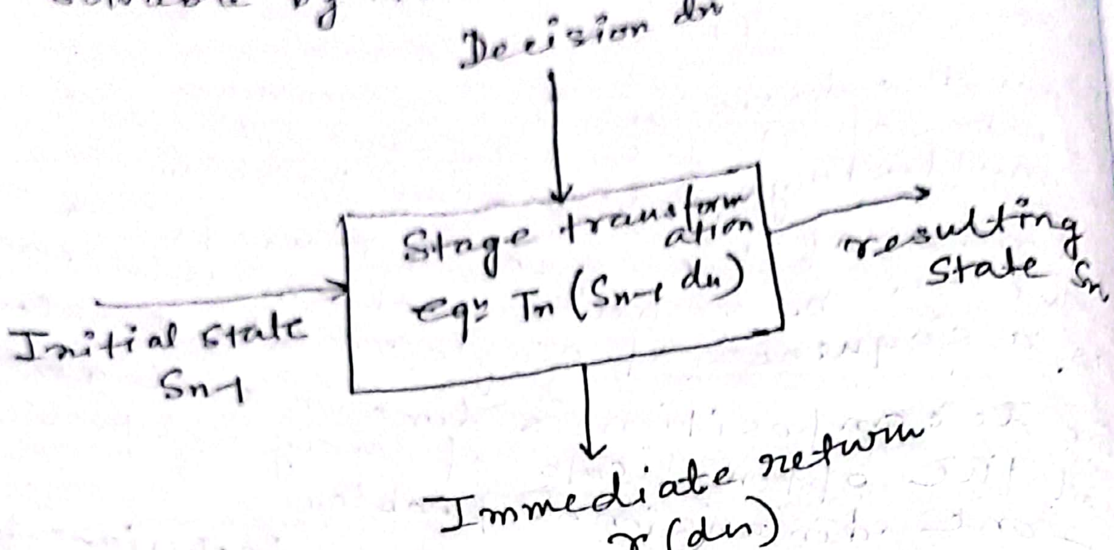
- In most practical problems decisions have to be made sequentially at different points in time, at different points in space for different subsystems.
- Since the decisions are to be made at number of stages, they are referred as multistage decision problem.
- In most of the cases, it is useful to solve an n-variable problem is represented as a sequence of single variable problem.
- The decomposition is done in such a way that the optimal soln of original problem n-one dimensional problem.
- Dynamic programming is a technique to deal with the optimization of multistage decision process. It was invented by American Mathematician Richard Bellman in the 1950s. Programming here means planning and dynamic is useful for problems where decision are taken in several distinct stages.

Characteristics:

- The problem can be divided into stages with a decision required at each stage. The stages may be certain time intervals or certain subdivision of problems for which independent decisions are possible

Each stage has a number of states associated with it. The variable that links the stages is the ~~stage~~ state variable.

• The decision at one stage transforms one state into a state in the next stage. The final stage must be solvable by itself.



Bellman's principle of optimality:

The dynamic programming method breaks this decision problem into smaller subproblems. Richard Bellman's principle of optimality describes how to do this.

Principle of optimality: An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision

Notations:

- At each stage, n of the dynamic program, there is:
  - a state variable, S_n.
  - an optimal decision variable, d_n
- For each value of S_n and d_n at stage n, there is
  - a return function value $r_n(d_n)$
- The output of the process at stage n is
  - the state variable for stage $n+1$, S_{n+1}.
  - S_{n+1} is calculated by a stage. Transformation fn, $T_{n+1}(S_n, d_{n+1})$
- The optimal value fn $f_n(S_n)$ is the cumulative return starting at the state S_n and proceeding to stage 1 under an optimal strategy.

Example:
Case 1: Single additive constraint, additively separable return.

Find u_i which minimizes

$$z = f_1(u_1) + f_2(u_2) + \cdots + f_n(u_n)$$

$$\text{s.t. } a_1 u_1 + a_2 u_2 + \cdots + a_n u_n \geq b$$

$$a_j, b \in R, \; a_j \geq 0, \; b > 0.$$

$$u_j \geq 0, \; j = 1, 2, \ldots n.$$

The objective or return fn z is a separable additive fn of the n variables u_j. $f_j(u_j)$ is a function of u_j only.

This is an n stage problem, the suffix j indicating the stage.

u_j are the decision variables.

With each decision u_j is associated a return f^n $f_j(u_j)$ which is a return at the jth stage.

Introduce state variables $x_1, x_2, \ldots x_n$ as:

$x_n = a_1u_1 + a_2u_2 + \ldots + a_nu_n \geq b$.

$x_{n-1} = a_1u_1 + a_2u_2 + \ldots + a_{n-1}u_{n-1} \geq x_n - a_nu_n$

$x_{n-2} = a_1u_1 + a_2u_2 + \ldots + a_{n-2}u_{n-2} \geq x_{n-1} - a_{n-1}u_{n-1}$

$- \quad - \quad -$

$x_1 = a_1u_1 = x_2 - a_2u_2$

Also, state transformation f^{ns} are

$x_{j-1} = t_j(x_j, u_j)$, $j = 1, 2, \ldots n$.

That is each state variable is a f^n of the next state and decision variables.

Since x_n is a function of all the decision variables, we may denote by $F_n(x_n)$ the minimum value of z for any feasible value of x_n.

$$F_n(x_n) = \min_{(u_1, u_2, \ldots u_n)} [f_1(u_1) + f_2(u_2) + \ldots + f_n(u_n)]$$

the minimization being over non negative values of u_j s.t. $x_n \geq b$.

Select a particular value of u_n and holding u_n fixed, minimize z over the remaining variables.

The minimum will be given by

$$f_n(u_n) = \min_{(u_1, u_2 \ldots u_{n-1})} [f_1(u_1) + f_2(u_2) + \ldots + f_{n-1}(u_{n-1})]$$

$$= f_n(u_n) + F_{n-1}(x_{n-1})$$

The values of $u_1, u_2 \ldots u_{n-1}$ which would make $[f_1(u_1) + f_2(u_2) + \ldots + f_{n-1}(u_{n-1})]$ minimum for a fixed u_n, thus depend upon x_{n-1} which in turn is a fn of x_n and u_n.

Also minimum z over all u_n, for any feasible x_n is

$$F_n(x_n) = \min_{u_n} (f_n(u_n) + F_{n-1}(x_{n-1})]$$

If somehow $F_{n-1}(x_{n-1})$ were known for ~~kn~~ all u_n, the above minimization would involve single variable u_n.

Repeating this argument, gives recursion formula as

$$F_j(x_j) = \min_{u_j} [f_j(u_j) + F_{j-1}(x_{j-1})], j = 2, 3, \ldots n$$

with $F_1(x_1) = f_1(u_1)$ and $x_{j-1} = t_j(x_j, u_j)$

is a Dynamic (Recursive) Programming Problem.