Dynamic Programming

1.0 HTC formulation for DP (Example 7E in W&W, pg. 242)

Consider a "mostly thermal" problem where we have one thermal plant and one hydro-reservior plant with the following data:

- Composite cost-rate function for thermal plant:
  $F(P_s) = 700 + 4.8P_s + 0.0005P_s^2$
  Units are \$/h
- Limits for thermal plant:
  $200 \le P_s \le 1200$
- Flow rate function for hydro plant
  $$q(P_H) = \begin{cases} 260 + 10P_H, & P_H > 0 \\ 0 & P_H = 0 \end{cases}$$
  Units are acre-ft/h
- Limits for hydro plant generation
  $0 \le P_H \le 200$
- Limits on flow rate
  $0 \le q \le 2260$
- Time interval of interest:
  24 hours, 4 hours per interval (i.e., n_j=4 hrs, j_{max}=6 intervals)
- Initial and final reservoir volumes:
  V_0=10,000 acre-ft
  V_6=10,000 acre-ft
- Inflow is r_j=1000 acre-ft/h for j=1,…,6
- Electric load is given in the following table:

Interval, j	$P_{load,j}$ (MW)
1	600
2	1000
3	900
4	500
5	400
6	300

To apply DP, we need to consider discrete states. Therefore we will assume a "volume storage step" of 2000 Acre-ft, i.e, the reservoir volume may only change in a single time interval by multiples of 2000Acre-ft.

From one time interval to the next, we may establish:

- a maximum volume increase corresponding to q=0 where the inflow of 1000 Acre-ft/hr occurs over 4 hours, giving a maximum volume increase of 4000Acre-ft.
- a maximum volume decrease corresponding to q=2260Acre-ft/hr over 4 hours, with 4000 Acre-ft of inflow over that 4 hours, results in a maximum volume decrease of

  2260*4-4000=5040Acre-ft

The initial volume together with the above stated maximum volume increases and decreases provides that we may describe the "feasible space" for the solution as a set of discrete points on a volume vs. time period graph, as shown below in Fig. 0a.

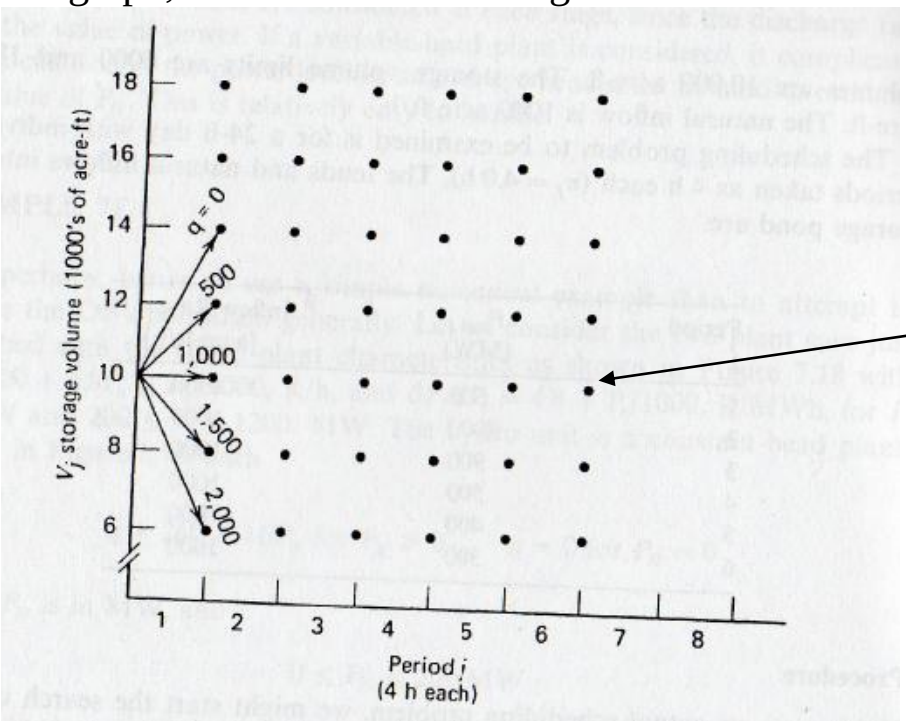

Fig. 0a

2.0 UC formulation for DP

Recall the example given in the file called "UC" where we considered a three-unit system over a 72 hour period. In that example, we identified three viable states and eight difference loading periods (including the initial loading period). Costs of transferring from each state in one period to the other states in the subsequent period were identified. The figure below was used to characterize these states and transition costs. We concluded that our problem was a shortest path problem – we desire to find the shortest path between the first and last node in the network.

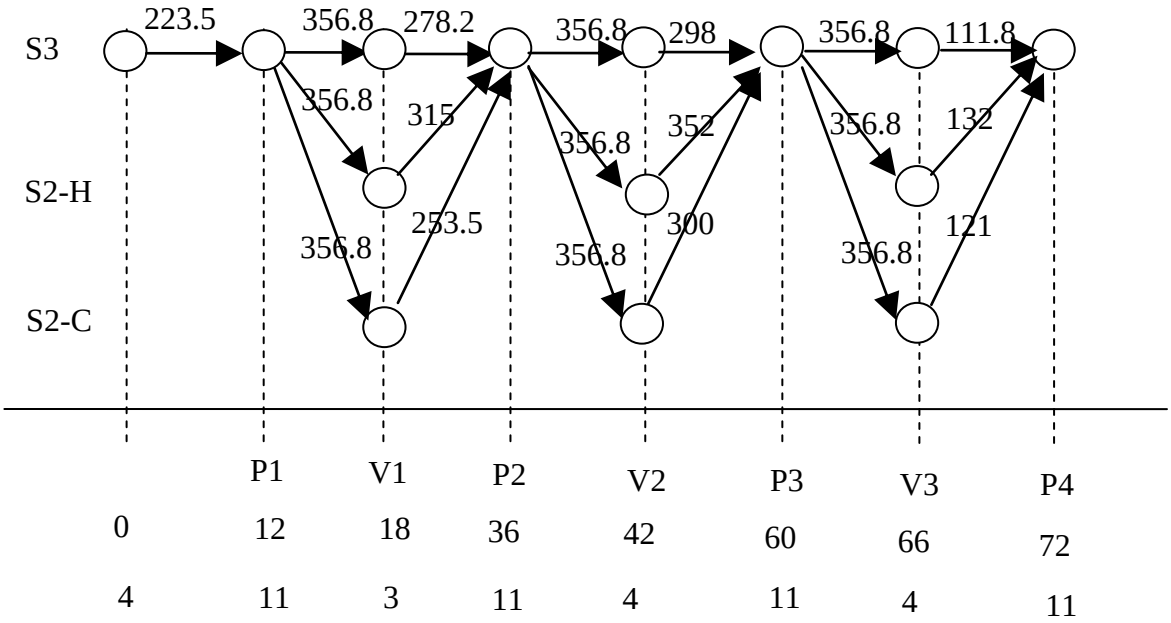

Fig. 0b

There are several algorithms for solving shortest path problems, including Dijkstra's algorithm, Dial's algorithm, Label correcting algorithms, All-pair algorithm, and dynamic programming (DP). We have also seen that it may be solved using branch and bound (B&B).

Although B&B is in many cases the most effective method, DP remains a competitive approach for some kinds of problems. There are many HTC codes today developed based on DP.

W&W treat DP in Appendix 3B of the economic dispatch chapter and describe its use for solving the economic dispatch problem with non-convex cost curves. W&W further use it in Section 5.2.2 of the UC chapter to solve the UC problem and again in Section 7.8 of the HTC chapter. We examine the basic algorithm in these notes following W&W's description of it their Appendix 3B.

3.0 Illustration of DP

Consider the following minimum cost problem, Fig. 0b. It is identical to the example given in W&W, Fig. 3.13, p. 75. The numbers k=1,2,3,4,5 represent hours. At the end of each hour, a decision is made regarding which state to enter for the next hour. These states are represented by circles. The arcs represent state transitions, and the numbers beside each arrow represent the transition cost. The objective is to find the minimum cost from A-N.

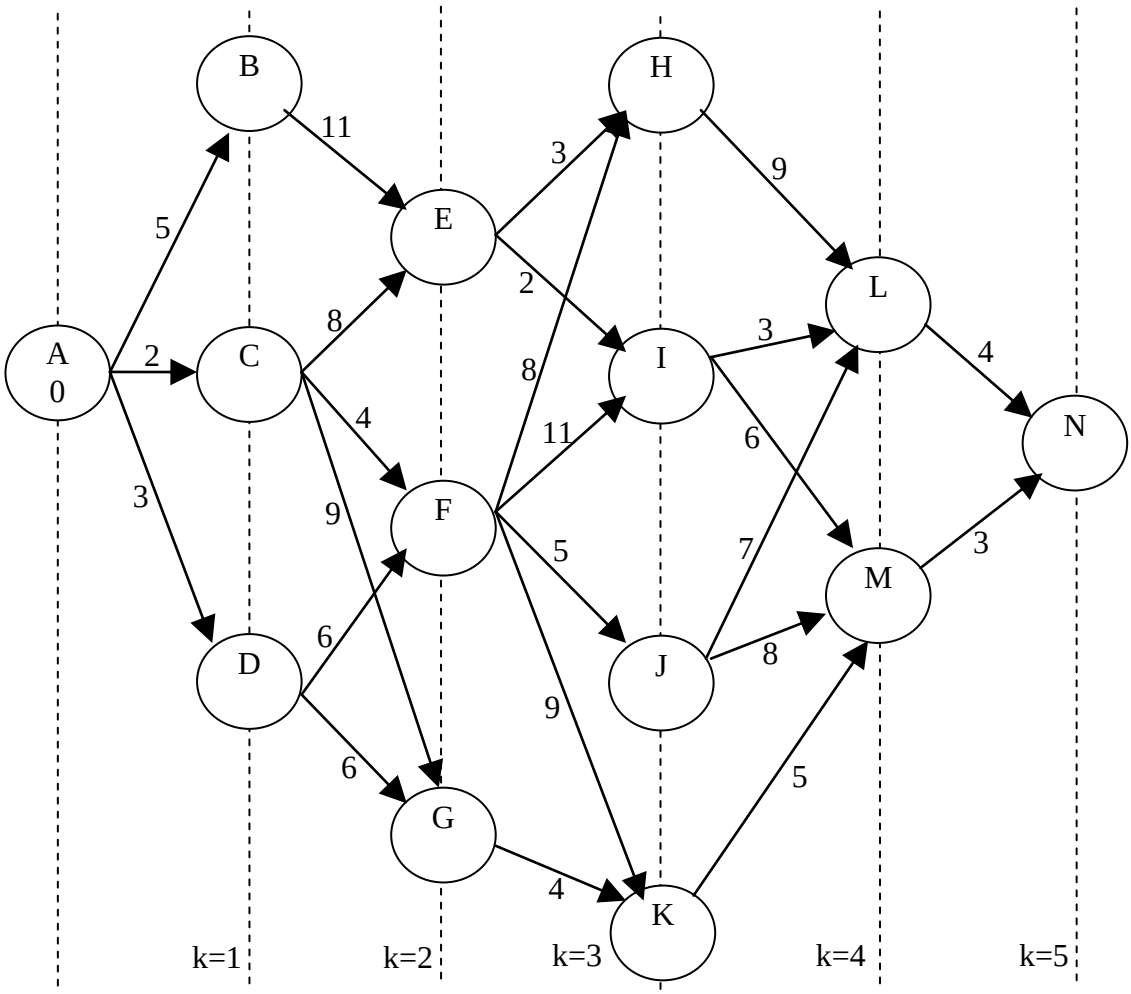

Fig. 0b

Observe that node A has a “zero” in the bottom of it. This indicates:

The minimum-cost sequence to
reach state A has cost of 0.

This is a trivial statement because A represents our starting hour, we are already “there,” and there is no cost to incur. But we will use this representation in the other circles as well. Remember that it will represent the minimum cost sequence of decisions from A to reach that node.

Our strategy is to compute the minimum cost sequence of decisions for every node.

➔Once we get the minimum cost sequence of decisions to reach the nodes at hour k=4, then the problem will be solved by finding the path that leads to node N at hour 5 for which the minimum cost at hour 4 plus the transition cost to node N is minimum. The resulting minimum total cost at node N can be expressed by:

$$F_{cost}(5,N) = \min_{m}\left\{F_{cost}(4,m) + S_{cost}(4,m:5,N)\right\} \quad (1)$$

where

- $F_{cost}(4,m)$ is the minimum total cost to reach node m at hour 4.
- $S_{cost}(4,m:5,N)$ is the cost of transition from node m in hour 4 to node N in hour 5.

Note that (1) minimizes by searching over m, that is, it finds the minimum cost to reach node N at hour 5 by searching over all of the nodes m in hour 4.

We use this notation repetitively in what follows, so we emphasize:

$$F_{cost}(5,N) = \min_{m}\left\{F_{cost}(4,m) + S_{cost}(4,m:5,N)\right\} \quad (1)$$

Min cost to reach node N, hr 5.	Min cost to reach node m, hr 4.	Cost of transition from node m, hr 4, to node N, hr 5..

So what we see that if we can obtain the minimum cost to reach each node (L and M) at hour 4, we can solve the problem using (1).

But now here is a new problem…. How to obtain the minimum cost to reach each node at hour 4?

We can treat each node at hour 4 just like we treated node N in the above example, i.e., we treat each node at hour 4 as if it were the terminal node. So consider node L, then we can find the minimum total cost to reach node L according to

$$F_{cost}(4,L) = \min_{m}\left\{F_{cost}(3,m) + S_{cost}(3,m:4,L)\right\} \quad (2a)$$

Likewise, for node M, we have

$$F_{cost}(4,M) = \min_m \{F_{cost}(3,m) + S_{cost}(3,m:4,M)\} \quad (2b)$$

Of course to solve (2a) and (2b), we need $F_{cost}(3,m)$ for nodes m=H,I,J,K. We do not have them, but we can obtain them using

$$F_{cost}(3,H) = \min_m \{F_{cost}(2,m) + S_{cost}(2,m:3,H)\} \quad (3a)$$

$$F_{cost}(3,I) = \min_m \{F_{cost}(2,m) + S_{cost}(2,m:3,I)\} \quad (3b)$$

$$F_{cost}(3,J) = \min_m \{F_{cost}(2,m) + S_{cost}(2,m:3,J)\} \quad (3c)$$

$$F_{cost}(3,K) = \min_m \{F_{cost}(2,m) + S_{cost}(2,m:3,K)\} \quad (3d)$$

And to solve (3a-3d), we need $F_{cost}(2,m)$ for nodes m=E,F,G. We do not have them, but we can obtain them using

$$F_{cost}(2,E) = \min_m \{F_{cost}(1,m) + S_{cost}(1,m:2,E)\} \quad (4a)$$

$$F_{cost}(2,F) = \min_m \{F_{cost}(1,m) + S_{cost}(1,m:2,F)\} \quad (4b)$$

$$F_{cost}(2,E) = \min_m \{F_{cost}(1,m) + S_{cost}(1,m:2,G)\} \quad (4c)$$

And to solve (4a-4c), we need $F_{cost}(1,m)$ for nodes m=B,C,D. We do not have them, but we can obtain them using

$$F_{cost}(1,B) = \min_m \{F_{cost}(0,m) + S_{cost}(0,m:1,B)\} \quad (5a)$$

$$F_{cost}(1,C) = \min_m \{F_{cost}(0,m) + S_{cost}(0,m:1,C)\} \quad (5b)$$

$$F_{cost}(1,D) = \min_m \{F_{cost}(0,m) + S_{cost}(0,m:1,D)\} \quad (5c)$$

To solve (5a-5c), we need $F_{cost}(0,m)$ for m=A. WE HAVE THAT!

We immediately see the strategy necessary for solving this problem. Since we know $F_{cost}(0,A)=0$, and we know all transition costs (the numbers on the arcs of Fig. 0b),

➔We can compute (5a), (5b), and (5c).
➔Then we compute (4a, 4b, 4c).
➔Then we compute (3a, 3b, 3c, 3d).
➔Then we compute (2a, 2b).
➔Then we compute (1), which solves our problem.

Let's execute the above procedure using our node-arc diagram to track where we are. We use a thicker line to mark the optimal path obtained to a node.

Computing (5a), (5b), and (5c):

$$F_{\mathrm{cost}}(1,B)=\min_{m}\left\{F_{\mathrm{cost}}(0,m)+S_{\mathrm{cost}}(0,m:1,B)\right\} \tag{5a}$$

$$F_{\mathrm{cost}}(1,C)=\min_{m}\left\{F_{\mathrm{cost}}(0,m)+S_{\mathrm{cost}}(0,m:1,C)\right\} \tag{5b}$$

$$F_{\mathrm{cost}}(1,D)=\min_{m}\left\{F_{\mathrm{cost}}(0,m)+S_{\mathrm{cost}}(0,m:1,D)\right\} \tag{5c}$$

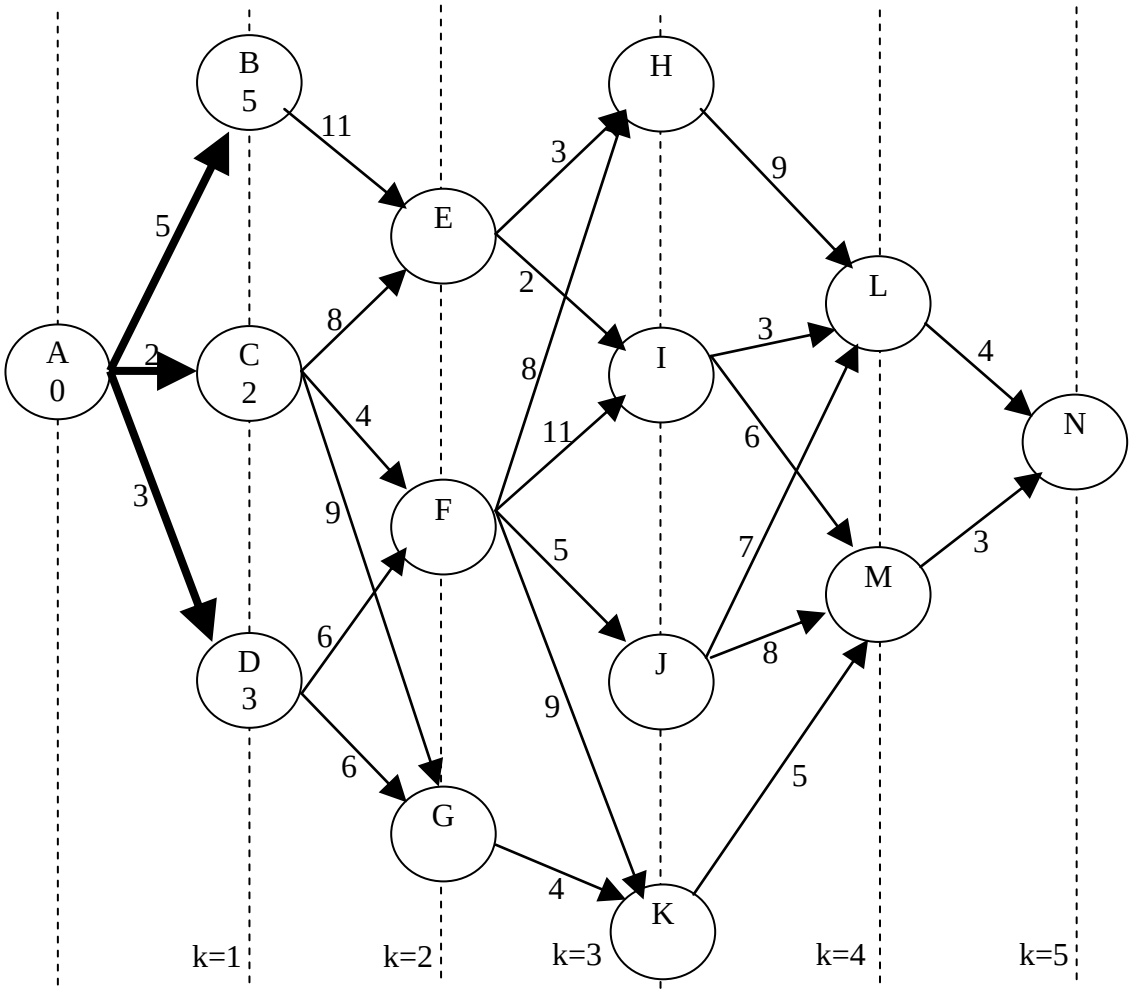

Fig. 1

Computing (4a, 4b, 4c):

$$F_{\text{cost}}(2,E) = \min_{m}\left\{F_{\text{cost}}(1,m) + S_{\text{cost}}(1,m:2,E)\right\} \quad (4a)$$

$$F_{\text{cost}}(2,F) = \min_{m}\left\{F_{\text{cost}}(1,m) + S_{\text{cost}}(1,m:2,F)\right\} \quad (4b)$$

$$F_{\text{cost}}(2,G) = \min_{m}\left\{F_{\text{cost}}(1,m) + S_{\text{cost}}(1,m:2,G)\right\} \quad (4c)$$

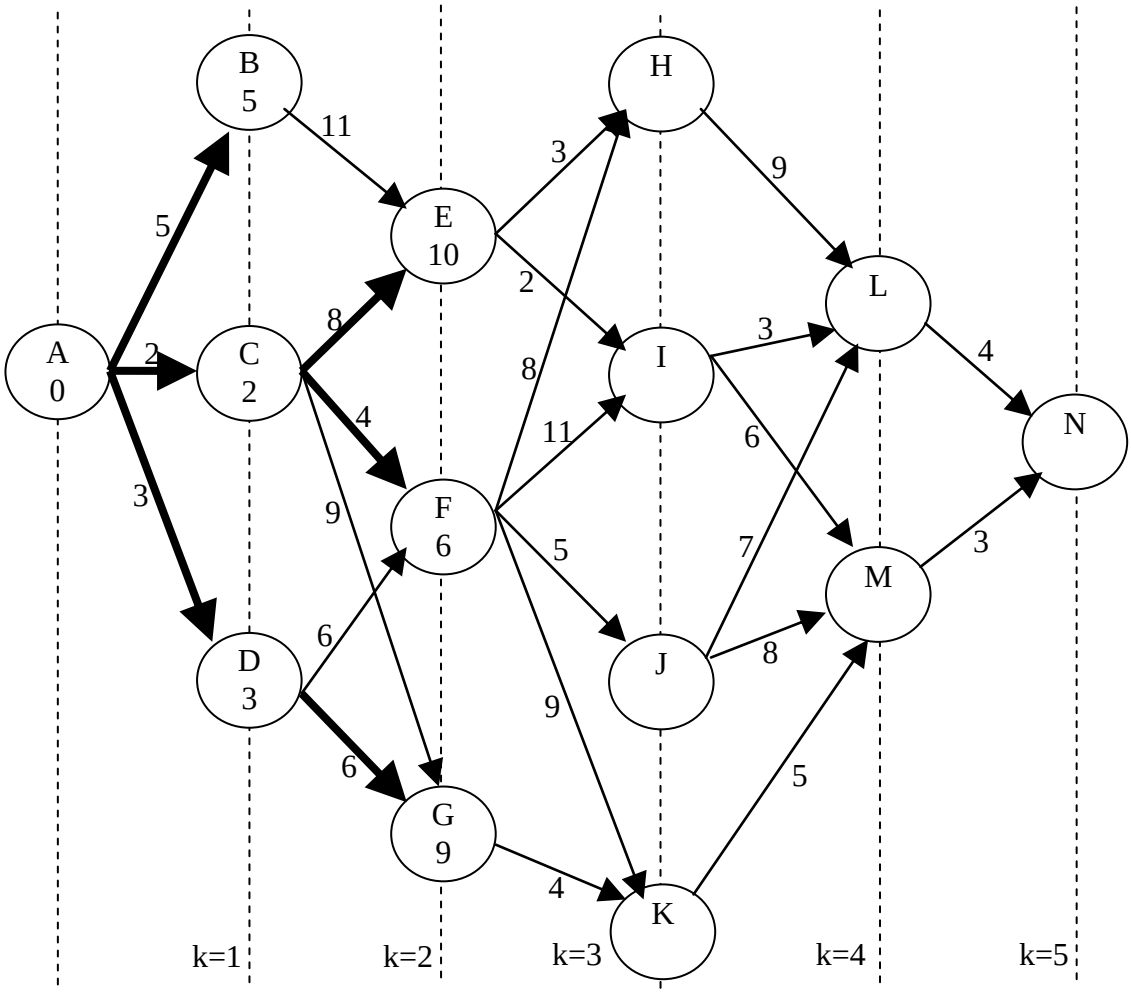

Fig. 2

Computing (3a), (3b), (3c), (3d):

$$F_{\text{cost}}(3,H) = \min_{m}\left\{F_{\text{cost}}(2,m) + S_{\text{cost}}(2,m:3,H)\right\} \quad (3a)$$

$$F_{\text{cost}}(3,I) = \min_{m}\left\{F_{\text{cost}}(2,m) + S_{\text{cost}}(2,m:3,I)\right\} \quad (3b)$$

$$F_{\text{cost}}(3,J) = \min_{m}\left\{F_{\text{cost}}(2,m) + S_{\text{cost}}(2,m:3,J)\right\} \quad (3c)$$

$$F_{\text{cost}}(3,K) = \min_{m}\left\{F_{\text{cost}}(2,m) + S_{\text{cost}}(2,m:3,K)\right\} \quad (3d)$$

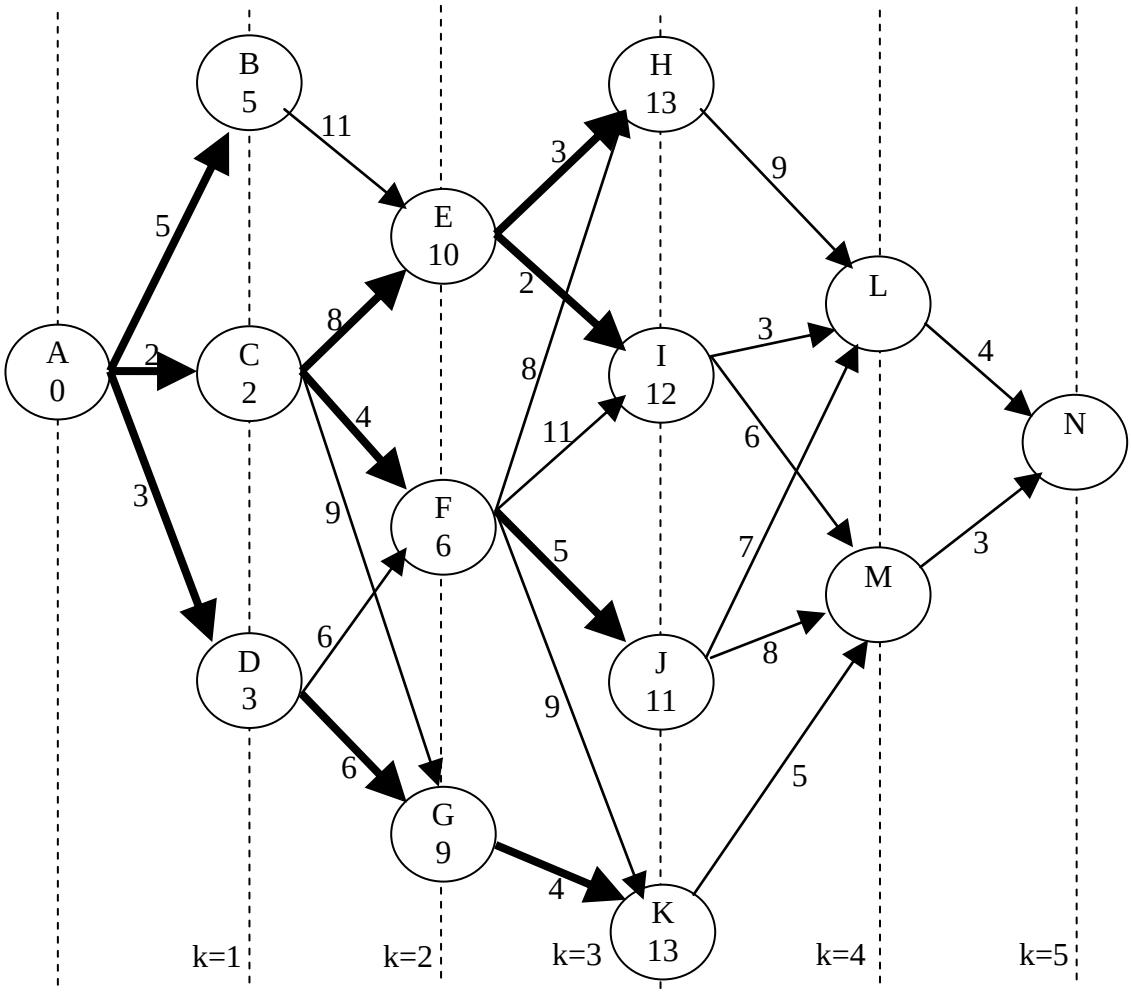

Fig. 3

Computing (2a, 2b):

$$F_{\text{cost}}(4,L) = \min_{m}\left\{F_{\text{cost}}(3,m) + S_{\text{cost}}(3,m:4,L)\right\} \tag{2a}$$

$$F_{\text{cost}}(4,M) = \min_{m}\left\{F_{\text{cost}}(3,m) + S_{\text{cost}}(3,m:4,M)\right\} \tag{2b}$$

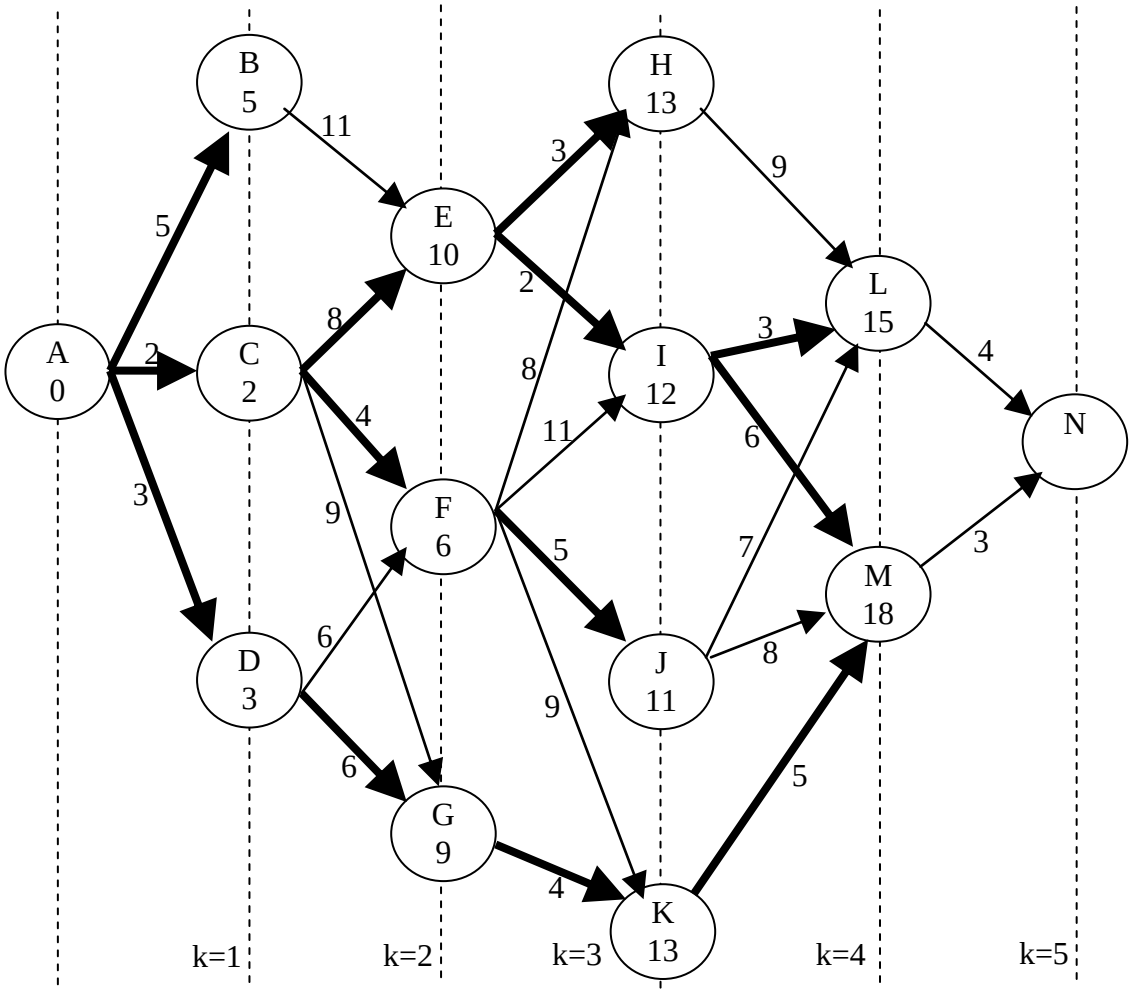

Fig. 4

Observe here that there are two solutions to (2b), i.e., in finding the minimum total cost to reach node M, we can either move through node I or node K; in each case, the minimum total cost is 18. We can record them both if we like, but it is unnecessary. As long as it only costs 18 to reach node M, we do not care what route we take to reach there.

Computing (1),

$$F_{\cos t}(5,N)=\min_{m}\left\{F_{\cos t}(4,m)+S_{\cos t}(4,m:5,N)\right\} \qquad (1)$$

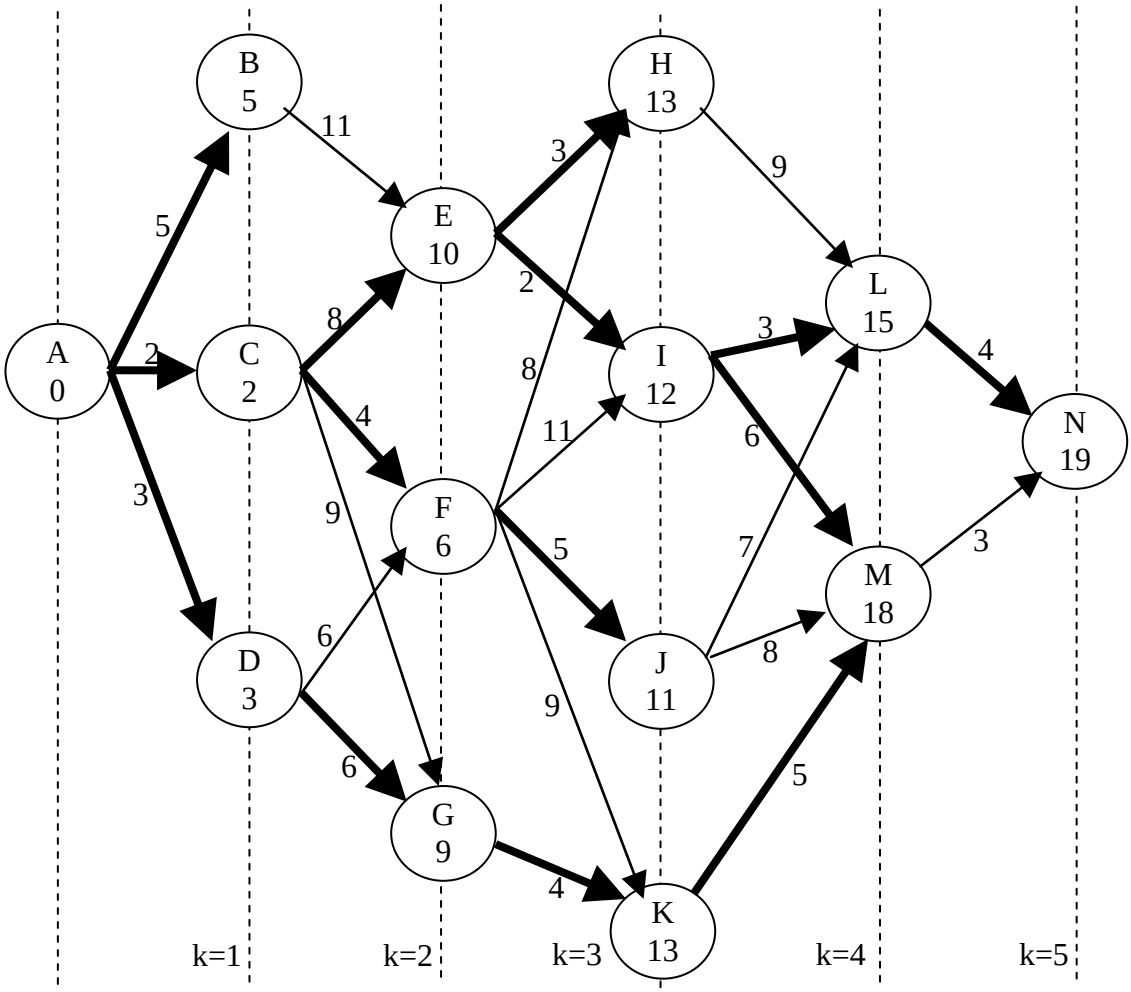

Fig. 5

From Fig. 5, we observe that the minimum cost is 19, and the optimal *policy* (a sequence of decisions) is given by path A-C-E-I-L-N.

4.0 Observations

We make several observations at this point.

1. When problems can be posed as a sequence of decisions, where
   - the sequence occurs in a defined manner from step k=0 to step k=1 to step k=2 to…
   - at each step, there are clearly defined states of existence (nodes)
   - transitions occur only between nodes in step k to nodes in step k+1
   - each transition can be assigned a value corresponding to what needs to be optimized in the problem

   then the problem can be solved by recursive use of;

   $$F_{\cos t}(k,n) = \min_{m}\left\{F_{\cos t}(k-1,m) + S_{\cos t}(k-1,m:k,n)\right\} \quad (6)$$

   Equation (6) is referred to as Bellman's equation, named for its inventor, Richard Bellman [1, 2].
2. The approach is called *dynamic programming* (DP). Our specific implementation of DP is forward DP. Forward DP works well when the initial condition is known, as is the case with most UC and HTC problems. It is also possible to formulate a backwards DP algorithm, appropriate when the terminal condition is known.
3. DP works for any problem where the following ***principle*** is satisfied:

   Version 1: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

   Version 2: A policy is optimal if, at a stated stage, whatever the preceding decisions may have been, the decisions still to be taken constitute an optimal policy when the result of the previous decisions is included.

   Version 3: An optimal policy must contain only optimal subpolicies.

Version 4: Every optimal solution to a problem contains optimal solutions to all subproblems.

This principle is called Bellman's *Principle of Optimality*.

Notice that the principle of optimality does not say:

If you have optimal solutions to all subproblems, then you can combine them to get an optimal solution.

With respect to our example problem examined above,

A-C-E-I-L-N	is the optimal solution to the problem of reaching N from A.
C-E-I-L-N	is the optimal solution to the problem of reaching N from C.
E-I-L-N	is the optimal solution to the problem of reaching N from E.
I-L-N	is the optimal solution to the problem of reaching N from I.
L-N	is the optimal solution to the problem of reaching N from L.

Therefore, our problem satisfies the principle of optimality.

Perhaps it is of some value to consider a case where the principle of optimality is not satisfied.

Consider a "longest path problem" for Fig. 6.

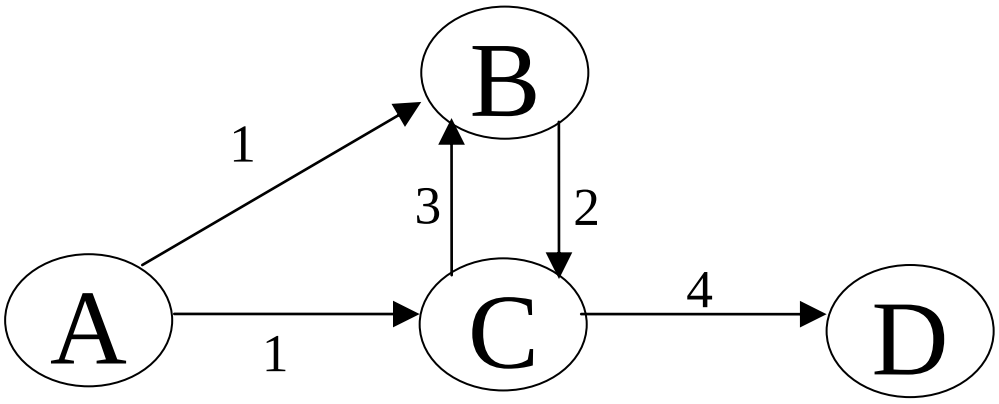

Fig. 6

Consider that each arc value represents time and that we want to maximize the time it takes us to get from A to D. But we can only touch each node once – this means our path can contain no cycles. The longest such path from A to D is A-B-C-D.

But what about the longest path from A to B? This is A-C-B.

Therefore the principle of optimality is not satisfied for this problem since the optimal solution to the problem (longest path from A to D) contains a non-optimal solution to a subproblem (longest path from A to B). Therefore this problem cannot be solved by DP!

---

[1] R. Bellman, "On the theory of dynamic programming," Proc. of the National Academy of Sciences, 1952.
[2] R. Bellman, "Dynamic Programming," Princeton University Press, Princeton, NJ, 1957.