

The Elementary Path Problem: A DP Solution

Clearly, the solution to the problem can be determined by a complete enumeration of all possible paths that begin from San Francisco (S. F.) and end at New York (N. Y.). For example, the total distance of the path

$$\text{San Francisco} \longrightarrow \text{Denver} \longrightarrow \text{Dallas} \longrightarrow \text{Atlanta} \longrightarrow \text{New York}$$

can be computed as $1274 + 874 + 779 + 885 = 3812$. Since the number of possible paths in this particular network is equal to 18 ($= 2 \times 3 \times 3$), the total computational effort of a complete enumeration is not unacceptable. However, as the number of cities in the network increases, the number of possible paths will grow very rapidly, making a complete enumeration prohibitively time consuming. It is therefore desirable to seek more efficient ways to determine the solution to our problem. Below, we will describe a dynamic programming solution.

The basic idea is to attempt to reduce the solution of this problem to the solution of a sequence of interrelated subproblems. With this in mind, let us imagine being at San Francisco now. Looking ahead, we see that there are two immediate choices for the first “stop” of the trip, namely Denver and Phoenix; and that the distances from S. F. to these two cities are 1274 and 755 miles, respectively. Clearly, this amount of information is not sufficient for us to arrive at the correct choice. Therefore, a natural question is: What is the minimal amount of additional information that will enable us to make a decision? A little bit of reflection now leads to the realization that if we have access to someone who can provide us with the shortest distances from Denver and Phoenix to New York, then we can simply add these distances to 1274 and 755, respectively, to determine which of Denver and Phoenix is the best first stop.

Since this “someone” is actually not available, what we have observed, in fact, is that the original problem of finding the shortest distance from S. F. to N. Y. can be reduced to the solution of two subproblems, that of finding the shortest distances from Denver and Phoenix to New York. Notice that these two subproblems are “smaller” than the original, in the sense that Denver and Phoenix are one-stop closer to N. Y. than San Francisco.

Continuation of this argument shows that to solve these two subproblems, we next need to know the shortest distances from Omaha, Oklahoma City, and Dallas to the final destination; and this, in turn, further implies that we need to determine the shortest distances from Columbus, Nashville, and Atlanta to the destination. It follows that the solution to our problem can be obtained by a recursive procedure that begins with these last three cities, which are only one leg away from N. Y., and works toward the starting location at San Francisco by building up a series of solutions to intermediate subproblems.

The outcome of an implementation of this recursive procedure is shown in Figure DP-2. Below, we will walk through a detailed explanation of the numbers shown in that figure.

The procedure begins with the three cities that are one leg away from the destination. Clearly, the best possible distances from these cities to N. Y. are simply given by their respective direct distances to New York. More formally, we shall, in general, refer to the best possible distance from any given city to N. Y. as the *optimal value* associated with that city. In terms of this language, what we have can be stated as:

The Optimal Value at Columbus = 535,

The Optimal Value at Nashville = 887,

and

The Optimal Value at Atlanta = 885.

These optimal values are reported in Figure DP-2 in the form of encircled numbers next to Columbus, Nashville, and Atlanta. In addition, notice that a small arrow that points toward N. Y. is drawn next to each of these cities; these arrows indicate the optimal decisions at these cities.

Next, we turn our attention to Omaha, Oklahoma City, and Dallas, which are two legs away from New York.

Looking ahead from Omaha, the next stop can be at Columbus, Nashville, or Atlanta. If the choice is Columbus, then the best possible distance to N. Y. equals the sum $798 + 535$, where the first number is the distance between Omaha and Columbus and the second number is the optimal value at Columbus. Similarly, if the choice is Nashville or Atlanta, then the corresponding best possible distance is given by $751 + 887$ or $994 + 885$. Since we are interested in minimizing total distance, the best choice should be the city that achieves the smallest of these three sums. Formally, this analysis can be summarized as:

$$\begin{aligned} \text{The Optimal Value at Omaha} &= \min \begin{bmatrix} \text{Columbus: } 798 + 535 \\ \text{Nashville: } 751 + 887 \\ \text{Atlanta: } 994 + 885 \end{bmatrix} \\ &= \min [1333, 1638, 1879] \\ &= 1333 \end{aligned}$$

(“min” means “the minimum of”) and the best choice at Omaha is to travel to Columbus. These results are reported in Figure DP-2 by the encircled entry 1333 and the arrow (pointing toward Columbus) adjacent to the Omaha node.

The analyses at Oklahoma City and at Dallas are similar. For Oklahoma City, we have:

$$\begin{aligned}
 \text{The Optimal Value at Oklahoma City} &= \min \begin{bmatrix} \text{Columbus: } 913 + 535 \\ \text{Nashville: } 683 + 887 \\ \text{Atlanta: } 854 + 885 \end{bmatrix} \\
 &= \min [1448, 1570, 1739] \\
 &= 1448
 \end{aligned}$$

and the best choice is to travel to Columbus. For Dallas, we have:

$$\begin{aligned}
 \text{The Optimal Value at Dallas} &= \min \begin{bmatrix} \text{Columbus: } 1040 + 535 \\ \text{Nashville: } 664 + 887 \\ \text{Atlanta: } 779 + 885 \end{bmatrix} \\
 &= \min [1575, 1551, 1664] \\
 &= 1551
 \end{aligned}$$

and the best choice is to travel to Nashville. These results are reported in Figure DP-2 in similar manners, next to the Oklahoma-City node and the Dallas node, respectively.

The next group of nodes consists of Denver and Phoenix. Both of these cities are three legs away from New York. For Denver, we have:

$$\begin{aligned}
 \text{The Optimal Value at Denver} &= \min \begin{bmatrix} \text{Omaha: } 538 + 1333 \\ \text{Oklahoma City: } 666 + 1448 \\ \text{Dallas: } 874 + 1551 \end{bmatrix} \\
 &= \min [1871, 2114, 2425] \\
 &= 1871
 \end{aligned}$$

and the best choice is to travel to Omaha. For Phoenix, we have

$$\begin{aligned}
 \text{The Optimal Value at Phoenix} &= \min \begin{bmatrix} \text{Omaha: } 1442 + 1333 \\ \text{Oklahoma City: } 1002 + 1448 \\ \text{Dallas: } 1083 + 1551 \end{bmatrix} \\
 &= \min [2775, 2450, 2634] \\
 &= 2450
 \end{aligned}$$

and the best choice is to travel to Oklahoma City. Again, these are reported in Figure DP-2 in similar manners.

We are finally ready for the decision at San Francisco. A similar comparison between Denver and Phoenix shows that:

$$\begin{aligned} \text{The Optimal Value at San Francisco} &= \min \begin{bmatrix} \text{Denver: } 1274 + 1871 \\ \text{Phoenix: } 755 + 2450 \end{bmatrix} \\ &= \min [3145, 3205] \\ &= 3145 \end{aligned}$$

and the best choice is to travel to Denver. As before, these are noted next to the San Francisco node in Figure DP-2.

In summary, we have arrived at the conclusion that the best possible total distance from San Francisco to New York is 3145 miles. Moreover, the sequence of optimal decisions is specified in Figure DP-2 by the stream of arrows that begins at San Francisco and ends at New York. In other words, the optimal path is:

$$\text{San Francisco} \longrightarrow \text{Denver} \longrightarrow \text{Omaha} \longrightarrow \text{Columbus} \longrightarrow \text{New York.}$$

This completes the solution of the problem.

What we will do next is to go through a careful reexamination of the solution process above. The purpose of this exercise is to formalize a number of general concepts that constitute the backbone of every dynamic-programming formulation. A solid understanding of these concepts, therefore, will prepare us for the solution of more-complicated problems.

The first concept is to think of the above network as being composed of a number of *stages*. Specifically, we can view the single node that represents San Francisco as stage 1, the pair of nodes that represent Denver and Phoenix as stage 2, the nodes that represent Omaha, Oklahoma City, and Dallas as stage 3, the nodes that represent Columbus, Nashville, and Atlanta as stage 4, and finally, the single node that represents New York as stage 5. In other words, in this example, stage 1 corresponds to the starting city, San Francisco, stage 2 consists of cities that are one leg away from S. F., stage 3 consists of cities that are two legs away from S. F., and stage 4 consists of cities that are three legs away from S. F., and stage 5 corresponds to the destination city, New York. This is depicted in Figure DP-3.

In most applications of dynamic programming, the stages will correspond to successive time periods.

Notice that in our problem, there is one decision to make, or one *action* to take, at every stage. (The actions correspond to traveling to particular cities.) This will also be the case in general. (Even when we have no decision to make, we can interpret it as taking the action of doing nothing.) Indeed, in many applications, the stages can also be defined according to the order of successive decisions or actions. We will encounter such examples in a later section.

The next concept is that each stage has a number of possible *states* associated with it. Simply put, information regarding the state can be viewed as a (further) specification of “where we are” within a stage. In this example, stage 1 has a single state, namely San Francisco; stage 2 has two possible states, namely Denver and Phoenix; stage 3 has three possible states, namely Omaha, Oklahoma City, and Dallas; stage 4 also has three possible states, namely Columbus, Nashville, and Atlanta; and stage 5 has a single state, namely New York. That is, each state within stage k , where $k = 1, 2, 3, 4, 5$, corresponds to a specific city that is $k - 1$ legs away from San Francisco.

In general, it is helpful to visualize the concepts of stages and states jointly as follows. Imagine a system whose status evolves over a given set of time periods according to a sequence of actions; and suppose a picture of this system is taken in every period. Then, each time period corresponds to a stage; and the picture taken in a particular stage corresponds to the state of the system in that stage.

Once the stages and states are appropriately defined, the primary task in the dynamic-programming solution of a problem is to formulate a general *recurrence relation* that relates the optimal value associated with each state within a given stage to the optimal values associated with states in the succeeding stage. The proper formulation of a recurrence relation is important because it is what enables us to compute the optimal values stage-by-stage until the optimal solution is found. Indeed, in the example above, we started with the optimal values at Columbus, Nashville, and Atlanta, which are in stage 4, and worked our way backward one stage at a time until the optimal solution, which is the optimal value at San Francisco, the only state in stage 1, is determined.

We now go through a careful formulation of the recurrence relation for our example. This formulation is somewhat abstract, but it will help us better understand the concept of a recurrence relation.

We begin with a little bit of notation. To facilitate cross referencing, we shall first rename the cities (or nodes) in the network. The idea is to number the cities within a stage from top to bottom and to refer to the i th city in stage k as node (i, k) . Thus, San Francisco becomes node $(1, 1)$, Denver becomes node $(1, 2)$, Phoenix becomes node $(2, 2)$, and so on. This is shown explicitly in Figure DP-4. Next, we shall denote by $d[(i, k), (j, k + 1)]$ the distance between nodes (i, k) and $(j, k + 1)$, where i and j are specific states within the consecutive stages k and $k + 1$, respectively. Thus, in our example, we have $d[(1, 1), (1, 2)] = 1274$, $d[(1, 1), (2, 2)] = 755$, $d[(1, 2), (1, 3)] = 538$, and so on. In slightly-more abstract terms, we shall also refer to $d[(i, k), (j, k + 1)]$ as the *one-stage cost* that is associated with taking “action j ” (i.e., the action of traveling to city j) at state i in stage k . Finally, let $V_k(i)$ denote the optimal value associated with node (i, k) ; conceptually, this means that we shall view $V_k(i)$ as a function, called the *optimal-value function*, over all possible stage and state combinations.

Now, imagine ourselves standing at node (i, k) , where i and k is any given pair of state and stage. Looking ahead from node (i, k) , the available actions correspond to visiting any one of the cities in the next stage. If the action taken is to travel to city j , then the best possible total cost (i.e., distance) from node (i, k) to the end (i.e., to node $(1, 5)$) must equal to the sum of the one-stage cost associated with traveling from node (i, k) to node $(j, k + 1)$ and the best possible total cost from node $(j, k + 1)$ to the end. In other words, the best possible total cost to the end associated with “action j ” is equal to the sum of the “immediate” one-stage cost associated with that action, $d[(i, k), (j, k + 1)]$, and the best possible total “future” cost from the resulting next node $(j, k + 1)$ to the end, $V_{k+1}(j)$. Since we are minimizing, it follows that $V_k(i)$ must equal to the smallest of the set of best possible total costs associated with the available actions at node (i, k) . This yields the recurrence relation

$$V_k(i) = \min_j \{d[(i, k), (j, k + 1)] + V_{k+1}(j)\},$$

where the minimization operation is over the set of all available actions (indexed by j) at state i in stage k .

As an explicit illustration of the recurrence relation, let us suppose $(i, k) = (2, 3)$, which corresponds to Oklahoma City. Then, with $j = 1$, we have $d[(i, k), (j, k + 1)] + V_{k+1}(j) = d[(2, 3), (1, 4)] + V_4(1)$; with $j = 2$, we have $d[(i, k), (j, k + 1)] + V_{k+1}(j) = d[(2, 3), (2, 4)] + V_4(2)$; and with $j = 3$, we have $d[(i, k), (j, k + 1)] + V_{k+1}(j) = d[(2, 3), (3, 4)] + V_4(3)$. It follows that

$$\begin{aligned} V_3(2) &= \min \{d[(2, 3), (1, 4)] + V_4(1), d[(2, 3), (2, 4)] + V_4(2), d[(2, 3), (3, 4)] + V_4(3)\} \\ &= \min \{913 + 535, 683 + 887, 854 + 885\} \\ &= \min \{1448, 1570, 1739\} \\ &= 1448 \end{aligned}$$

and the optimal value 1448 is achieved by taking action 1. Indeed, this calculation corresponds precisely to the earlier one for Oklahoma City.

That the best possible total cost associated with action j *must* equal to the sum of $d[(i, k), (j, k + 1)]$ and $V_{k+1}(j)$ is called the *principle of optimality*. This is a consequence of the obvious fact that regardless which specific action is taken at node (i, k) (and, in fact, at all earlier nodes), one must proceed optimally from the next node to the end.

With the recurrence relation in place, the final phase of the solution procedure consists of the recursive computation of the $V_k(i)$'s.

In the earlier calculations, we started the recursion with $V_4(1) = 535$, $V_4(2) = 887$, and $V_4(3) = 885$. Such a set of initial optimal values is called the *boundary condition*. Interestingly, note that we could also have started with the simpler-looking boundary condition

$V_5(1) = 0$ (this, in fact, was indicated in Figure DP-2 by the encircle entry 0 next to New York), which means that if we are already at New York, then there is no further cost. It is easy to see that there is little difference between these two boundary conditions in terms of overall computational effort.

The remainder of the recursive computation can be presented in the form of a series of tables, one for each stage. This is done as follows.

For stage 3, the calculations are described in the table below.

Stage 3:

i	$d[(i, 3), (j, 4)] + V_4(j)$			$V_3(i)$	$j_3^*(i)$
	$j = 1$	$j = 2$	$j = 3$		
1	798 + 535	751 + 887	994 + 885	1333	1
2	913 + 535	683 + 887	854 + 885	1448	1
3	1040 + 535	664 + 887	779 + 885	1551	2

Thus, the first column lists all possible states in stage 3; the next 3 columns list the details of the evaluation of $d[(i, 3), (j, 4)] + V_4(j)$ for all combinations of i and j (i.e., state and action); the fifth column lists $V_3(i)$ for all i , which are equal to the row-wise minimums of the entries in the three previous columns; and the final column lists, for every state i , the optimal action, denoted by $j_3^*(i)$, that achieves $V_3(i)$.

Similarly, the calculations for stage 2 are described in the next table.

Stage 2:

i	$d[(i, 2), (j, 3)] + V_3(j)$			$V_2(i)$	$j_2^*(i)$
	$j = 1$	$j = 2$	$j = 3$		
1	538 + 1333	666 + 1448	874 + 1551	1871	1
2	1442 + 1333	1002 + 1448	1083 + 1551	2450	2

Note that there are only two states in stage 2.

In stage 1, there is only a single state. The calculations for this stage are shown in the table below.

Stage 1:

i	$d[(i, 1), (j, 2)] + V_2(j)$		$V_1(i)$	$j_1^*(i)$
	$j = 1$	$j = 2$		
1	1274 + 1871	755 + 2450	3145	1

Since $V_1(1) = 3145$, we conclude that the best possible total distance from San Francisco to New York is 3145 miles. Moreover, the sequence of optimal actions can be read from these tables sequentially as follows. From the stage-1 table, we have $j_1^*(1) = 1$, which means we should travel from S. F. to Denver. Since Denver corresponds to state 1 in stage 2, we next look up $j_2^*(1)$ in the stage-2 table; since $j_2^*(1) = 1$, we should, therefore, travel from Denver

to Omaha. Since Omaha corresponds to state 1 in stage 3, an inspection of the first row in the stage-3 table again shows that $j_3^*(1) = 1$; therefore, we should next travel from Omaha to Columbus. Finally, from Columbus, the only possible action is to travel directly to New York, as specified by the boundary condition.

You should now verify that the just-completed recursive computation is identical to the one presented in Figure DP-2.

Finally, note that if we opted to adopt the boundary condition $V_5(1) = 0$, then an additional table would be necessary. Specifically, for stage 4, we would have the table below.

Stage 4:	i	$d[(i, 4), (1, 5)] + V_5(1)$	$V_4(i)$	$j_4^*(i)$
	1	$535 + 0$	535	1
	2	$887 + 0$	887	1
	3	$885 + 0$	885	1

This should be inserted before the stage-3 table.