

The Project-Planning Problem

Consider a project that has N essential components. The project is a success only if all N components are successful. The implementation of each component is to be assigned to a team of workers; and the probability of a successful implementation of a component is a function of the size of the team assigned to it. Given that there is a total of w workers available for assignment, the problem of interest is to determine an allocation of the workers that maximizes the success probability of the entire project. We shall refer to this problem as the *project-planning problem*.

This problem is similar in spirit to the knapsack problem, in that we are interested in an optimal allocation of a limited resource. Notice, however, that the objective function in the project-planning problem is *multiplicative*, whereas the objective function in the knapsack problem is additive. This means that the success probability of the project is the product of the success probabilities of its components.

Formally, let

$p_k(j)$ = the success probability of component k if j workers are assigned to it, for $k = 1, 2, \dots, N$;

and suppose that these probabilities are given. Then, the project-planning problem can be formulated as:

$$\begin{array}{ll} \text{Maximize} & p_1(x_1) \times p_2(x_2) \times \cdots \times p_N(x_N) \\ \text{Subject to:} & \\ & \sum_{k=1}^N x_k \leq w, \end{array}$$

where x_1, x_2, \dots, x_N are nonnegative integer-valued decision variables, defined by

x_k = the size of the team assigned to component k , for $k = 1, 2, \dots, N$.

Notice that if we assume, which we will, that the $p_k(j)$'s are nondecreasing in j for every k , then the inequality in this formulation can be replaced by an equality.

As a simple numerical example, suppose $N = 3$, $w = 5$, and the $p_k(j)$'s are given in the tables below.

j	$p_1(j)$	j	$p_2(j)$	j	$p_3(j)$
0	0	0	0	0	0
1	0.5	1	0.4	1	0.3
2	0.6	2	0.7	2	0.6
3	0.7	3	0.8	3	0.8
4	0.8	4	0.9	4	0.85
5	0.85	5	0.95	5	0.85

Observe that these $p_k(j)$'s are nondecreasing in j for every k . Moreover, since $p_k(0) = 0$ for every k , at least one worker should be assigned to each component to ensure a positive success probability for the project. As a sample allocation of workers, let $x_1 = 2$, $x_2 = 1$, and $x_3 = 2$. Since $p_1(2) = 0.6$, $p_2(1) = 0.4$, and $p_3(2) = 0.6$, this allocation has a success probability $0.6 \times 0.4 \times 0.6 = 0.144$.

We now describe how to derive the optimal allocation using dynamic programming. As usual, the solution procedure will be in four steps. Since the procedure is very similar to that for the knapsack problem, we will be brief.

Stages and States

Since there is one decision for each component, we define one stage for each component, independent of the order of the components.

Suppose that decisions for components 1 through $k - 1$ have already been made, and that we are in charge of making the remaining decisions for components k through N . Then, the answer to the consultant question is that we need to know the number of remaining, or available, workers at the beginning of stage k . We will, therefore, define the number of remaining workers as the state.

Optimal-Value Function

In the language of the present problem, let

$V_k(i)$ = the best possible probability for successful implementations of components k through N , assuming that the number of remaining workers is i .

Our goal is to determine $V_1(w)$; in the simple numerical example above, this means that we are interested in $V_1(5)$.

Recurrence Relation

Suppose we are now in stage k ; and suppose further that we are in state i , which means that the number of remaining workers is i . Recall that the project cannot succeed unless at least one worker is assigned to every component. (We assume $w \geq N$; otherwise, the problem is not meaningful.) Hence, the fact that we are in stage k implies that at least one worker should have been assigned to each of components 1 through $k - 1$. It follows that we expect i to be less than or equal to $w - (k - 1)$. Moreover, since we should reserve at least one worker for each of the remaining components, we also expect i to be at least $N - (k - 1)$. In other words, if either of these two conditions is not met, then we must have $V_k(i) = 0$. We will, therefore, only consider i 's that satisfy $N - k + 1 \leq i \leq w - k + 1$.

Consider any given i , where $N - k + 1 \leq i \leq w - k + 1$. As noted above, the value of x_k should be at least one. Moreover, observe that after assigning x_k workers to component k , the number of workers that will be available for assignment to components $k + 1$ through N will be $i - x_k$; that is, the state in the next stage, namely stage $k + 1$, will be $i - x_k$. It follows that the value of x_k should also be such that $i - x_k \geq N - k$. Note that this inequality is equivalent to $x_k \leq i - N + k$. Therefore, the feasible range for x_k is $1 \leq x_k \leq i - N + k$.

For any x_k in the just-specified range, the probability for component k to be a success is $p_k(x_k)$. Since the subsequent state in stage $k + 1$ will be $i - x_k$, the best possible probability for the successful implementation of all remaining components is, by definition, equal to $V_{k+1}(i - x_k)$. It follows that if we assign x_k workers to component k , i.e., if we take action x_k , then the best possible probability for the successful implementations of components k through N is equal to $p_k(x_k)V_{k+1}(i - x_k)$.

Combining the discussions above now yields the following recurrence relation:

$$V_k(i) = \max_{1 \leq x_k \leq i - N + k} [p_k(x_k) \times V_{k+1}(i - x_k)].$$

We next move on to the recursive computation of the $V_k(i)$'s.

Computation

We will illustrate the computation with the simple numerical example above. With $N = 3$, the total number of stages is 3.

Consider stage 3. Our discussion above shows that the relevant states are from $N - k + 1$ to $w - k + 1$, i.e., from $3 - 3 + 1 = 1$ to $5 - 3 + 1 = 3$. Clearly, for any state i in this range, we should let $x_3 = i$ to maximize the success probability of component 3; that is, we have $V_3(i) = p_3(i)$. This yields the boundary condition specified in the table below.

Stage 3:	i	$V_3(i)$	x_3^*
	1	0.3	1
	2	0.6	2
	3	0.8	3

We now consider stage 2, where the relevant states are from $N - k + 1$ to $w - k + 1$, i.e., from $3 - 2 + 1 = 2$ to $5 - 2 + 1 = 4$. For state 2, the value of x_2 can be from 1 to $i - N + k = 2 - 3 + 2 = 1$. Thus, the only feasible action is to let $x_2 = 1$, which implies that

$$\begin{aligned} V_2(2) &= \max_{1 \leq x_2 \leq 1} [p_2(x_2) \times V_3(2 - x_2)] \\ &= p_2(1) \times V_3(1) \\ &= 0.4 \times 0.3 \\ &= 0.12, \end{aligned}$$

where $V_3(1) = 0.3$ is taken from the stage-3 table. For state 3, the value of x_2 can be from 1 to $i - N + k = 3 - 3 + 2 = 2$. Thus, the feasible actions are $x_2 = 1$ and $x_2 = 2$; therefore, the recurrence relation evaluates to

$$\begin{aligned}
V_2(3) &= \max_{1 \leq x_2 \leq 2} [p_2(x_2) \times V_3(3 - x_2)] \\
&= \max [p_2(1) \times V_3(2), p_2(2) \times V_3(1)] \\
&= \max [0.4 \times 0.6, 0.7 \times 0.3] \\
&= \max [0.24, 0.21] \\
&= 0.24
\end{aligned}$$

and the optimal action is to let $x_2 = 1$. Finally, for state 4, the value of x_2 can be from 1 to $i - N + k = 4 - 3 + 2 = 3$. Thus, the feasible actions are $x_2 = 1$, $x_2 = 2$, and $x_2 = 3$; therefore, the recurrence relation evaluates to

$$\begin{aligned}
V_2(4) &= \max_{1 \leq x_2 \leq 3} [p_2(x_2) \times V_3(4 - x_2)] \\
&= \max [p_2(1) \times V_3(3), p_2(2) \times V_3(2), p_2(3) \times V_3(1)] \\
&= \max [0.4 \times 0.8, 0.7 \times 0.6, 0.8 \times 0.3] \\
&= \max [0.32, 0.42, 0.24] \\
&= 0.42
\end{aligned}$$

and the optimal action is to let $x_2 = 2$. These calculations are summarized in the stage-2 table below.

Stage 2: i	Actions			$V_2(i)$	x_2^*
	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$		
2	0.4×0.3	—	—	0.12	1
3	0.4×0.6	0.7×0.3	—	0.24	1
4	0.4×0.8	0.7×0.6	0.8×0.3	0.42	2

Finally, consider stage 1, where the only state is 5. Since $i - N + k = 5 - 3 + 1 = 3$, the feasible actions are $x_1 = 1$, $x_1 = 2$, and $x_1 = 3$; therefore, the optimal value for state 5 is

$$\begin{aligned}
V_1(5) &= \max_{1 \leq x_1 \leq 3} [p_1(x_1) \times V_2(5 - x_1)] \\
&= \max [p_1(1) \times V_2(4), p_1(2) \times V_2(3), p_1(3) \times V_2(2)] \\
&= \max [0.5 \times 0.42, 0.6 \times 0.24, 0.7 \times 0.12] \\
&= \max [0.21, 0.144, 0.084] \\
&= 0.21
\end{aligned}$$

and the optimal action is to let $x_1 = 1$. This yields the table below.

Stage 1: i	Actions			$V_1(i)$	x_1^*
	$x_1 = 1$	$x_1 = 2$	$x_1 = 3$		
5	0.5×0.42	0.6×0.24	0.7×0.12	0.21	1

Since $V_1(5) = 0.21$, we conclude that the best success probability for the project is 0.21.

The sequence of optimal actions can be read from the above tables sequentially. From the stage-1 table, we have $x_1^* = 1$. Since $x_1^* = 1$, the subsequent state at stage 2 will be $5 - 1 = 4$; therefore, we should next look up the row with $i = 4$ in the stage-2 table, which shows that $x_2^* = 2$. This, in turn, implies that the subsequent state at stage 3 will be $4 - 2 = 2$; and a reading of the row with $i = 2$ in the stage-3 table shows that $x_3^* = 2$. Thus, the optimal allocation is to assign 1 worker to component 1, 2 workers to component 2, and 2 workers to component 3. This completes the solution of the numerical example.

Discussion

As usual, the recursive computation above can also be carried out via a network representation of the problem. Since this is very similar to what we did for the knapsack problem, we will simply display the outcome of the calculations in Figure DP-7. The only difference is that the optimal values are determined by the multiplication, as opposed to the addition, of a one-stage probability and a corresponding probability at the next stage. You should verify that the optimal values in Figure DP-7 are identical to those presented in the series of tables above.

Our solution procedure can be easily adapted to handle other objective functions. As an example, suppose our objective is to maximize the probability of having at least one successful component. That is, consider the objective function

$$\text{Maximize} \quad 1 - [1 - p_1(x_1)] \times [1 - p_2(x_2)] \times \cdots \times [1 - p_N(x_N)].$$

Clearly, this is equivalent to

$$\text{Minimize} \quad [1 - p_1(x_1)] \times [1 - p_2(x_2)] \times \cdots \times [1 - p_N(x_N)],$$

which assumes the same multiplicative form as the original formulation. It follows that essentially the same procedure can be applied to solve this new problem.