Dynamic Programming Recitation

Dynamic programming is a methodology to approach optimization problems that involve a non-convex feasible region. Each problem needs to be translated into the formalism developed for the methodology, which involves stages, states, return function, and cumulative return function.

$g_i(X_i)$ - return function when state X is applied to stage i
Stage – time (periods), space (locations, travel), sequence (logistics), portfolio of investments, redundant components in a system
State X – set of possible choices for each stage (used in individual return function)
K – one of the possible states (used in cumulative return function)
$f_i(K)$ - cumulative return function: what is best way to "be in" or "arrive at" state (K) over first (i) stages?

Both stages and states are sets. DP is applicable both when the return function is continuous and when it is discontinuous (step-wise). The recurrence formula gives the cumulative return function; it does not consider constraints.

Questions to ask when approaching a problem:
How can I best do __x__ in stage 1?
How can I best do __x__ in stage 1 & 2?
How can I best do __x__ in stage 1, 2, ... & i?

Exercise 7.13 (modified)

Consider a truck whose maximum loading capacity is 10 tons. Suppose that there are 3 different items, various quantities of which are to make up a load to be carried to a remote geographical area. Suppose we wish to maximize the value of what the truck carries to the inhabitants, given the following weights and values of the four items.

Item	Weight	Value
1	3 tons	$5
2	4 tons	$7
3	5 tons	$8

a) Find the optimal solution by dynamic programming.
b) Write the recurrence formula for this problem
c) Define the assumptions of dynamic programming.

Solution

a) This is a non-sequential problem, so it does not matter how the stages are organized. Here the three stages are taken as each item category that we can load on the truck. There are three stages (i = 1, 2, 3). In stage 1, we consider loading only item 1. In stage 2, we consider loading items 1 and 2, and in stage 3, items 1, 2, and 3. The states (X_i) are the

amount of units of each item loaded on the truck. The return function $g_i(X_i)$ gives the total value in $ that can be transported, given the physical constraint of 10 tons the truck can carry. This constraint also limits the total value we can get by combining different items in different amounts.

The following table shows the return function depending on the amount of each item as if loaded separately. The value is determined by multiplying the number of units of each item and the value associated with it.

Amount of item i	Return function $g_i(X_i)$		
	Stage i = 1	Stage i = 2	Stage i = 3
0	$0 (0 tons)	$0 (0 tons)	$0 (0 tons)
1	$5 (3 tons)	$7 (4 tons)	$8 (5 tons)
2	$10 (6 tons)	$14 (8 tons)	$16 (10 tons)
3	$15 (9 tons)	Beyond 10 tons constraint	Beyond 10 tons constraint

The cumulative return function is:

$$f_i(K) = \max[g_i(X_i) + f_{i-1}(K-X_i)].$$

Note that this function is particular to this optimization problem, and varies from one problem to the other. In some cases, the function may involve minimization of a given quantity, so the cumulative return function is min[*]. In this problem, this function means that for a given number of items loaded on board, we find the maximum value we can obtain by loading a certain number of units of a given item (up at the given stage), given the maximum value obtained by loading units of the previous item. For each stage, we evaluate all possible amounts of units that can be loaded on board, subject to the 10 tons constraint. Since no more than 3 units of any item can be loaded, K = 0, 1, 2, or 3.

For the first stage, the cumulative function is equal to the return function: $f_1(K) = g_1(K)$

$f_1(0)$ = $0 (0 tons) $f_1(1)$ = $5 (3 tons) $f_1(2)$ = $10 (6 tons) $f_1(3)$ = $15 (9 tons)

For this stage, there is only one way of boarding 0, 1, 2, or 3 units of item 1.

For the second stage, the cumulative function is: $f_2(K) = \max[g_2(X_2) + f_1(K-X_2)]$

$f_2(0) = \max (g_2(0) + f_1(0 - 0) = \$0$

- ⇨ There is only way to get $0 by combining the 2 first items, which is by having none of both.

$f_2(1) = \max [g_2(1) + f_1(0), g_2(0) + f_1(1)] = \max[\$7 + \$0, \$0 + \$5] = \$7$ (4 tons)

- ⇨ Here, X_2 takes all possible states of item 2 on board the truck, which is either 0 or 1. K = 1 remains the same in our evaluation of the possible combinations of having a total of 1 unit on board.
- ⇨ The possible ways of having 1 item on board is to have 1 unit of item 1 and 0 unit of item 2, or 0 unit of item 1 and 1 unit of item 2.
- ⇨ The most value is obtained by having 1 unit of item 2, which weighs 4 tons and provides \$7 of value.

$f_2(2)$ = max[$g_2(2) + f_1(0)$, $g_2(1) + f_1(1)$, $g_2(0) + f_1(2)$]
= max[\$14 + 0, \$7 + \$5, \$0 + \$10]
= \$14 (8 tons, 2 units of item 2)

- ⇨ This lists all the possibilities of having 2 units on board the truck. The most value is obtained by having 2 units of item 2, with a weight of 8 tons and \$14.

$f_2(3)$ = max[$g_2(3) + f_1(0)$, $g_2(2) + f_1(1)$, $g_2(1) + f_1(2)$, $g_2(0) + f_1(3)$]
= max[\$0 (> 10 tons), \$0 (> 10 tons), \$7 + \$10 (10 tons), \$0 + \$15 (9 tons]
= \$17 (10 tons, 1 unit of item 2 and 2 units of item 1)

For the third stage, the cumulative function is: $f_3(K) = \max[g_3(X_3) + f_2(K-X_3)]$

$f_3(0)$ = max ($g_3(0) + f_2(0)$) = \$0

$f_3(1)$ = max [$g_3(1) + f_2(0)$, $g_3(0) + f_2(1)$]
= max[\$8 + \$0, \$0 + \$7]
= \$8 (5 tons, 1 unit of item 3)

$f_3(2)$ = max[$g_3(2) + f_2(0)$, $g_3(1) + f_2(1)$, $g_3(0) + f_2(2)$]
= max[\$16 + 0, \$8 + \$7, \$0 + \$14]
= \$16 (10 tons, 2 units of item 3)

$f_3(3)$ = max[$g_3(3) + f_2(0)$, $g_3(2) + f_2(1)$, $g_3(1) + f_2(2)$, $g_3(0) + f_2(3)$]
= max[\$0 (> 10 tons), \$0 (> 10 tons), \$0 (> 10 tons), \$0 + \$17 (10 tons]
= \$17 (10 tons, 1 unit of item 2 and 2 units of item 1)

Therefore, the best policy is to take 2 units of item 1 (\$10, 6 tons), and 1 unit of item 2 (\$7, 4 tons) for a total maximum value of \$17 and 10 tons load.

b) As mentioned above, $f_i(K) = \max[g_i(X_i) + f_{i-1}(K-X_i)]$.

c) The assumptions for dynamic programming are:
a. monotonicity, such that improvements in each return function lead to improvements in the objective function, and
b. separability, so that each return function is independent.

How does dynamic programming reduce the number of combinations to evaluate?

Have a look at the decision tree in Excel file DPtree-7.13.xls. The yellow paths are those that are explored in the dynamic programming process, the black ones are pruned out in the process. Those paths correspond to the cumulative return functions explored through the process. Note that other paths are systematically pruned out and not shown in the tree: the paths providing a load higher than the 10 tons constraint. Therefore, dynamic programming is very efficient at reducing the number of possible paths to explore in a given combinatorial problem.

How does dynamic programming relate to the decision analysis performed using the binomial model of copper price?

In the binomial lattice decision analysis example, the stages, levels, return functions, and cumulative return functions can be considered as follows:

Stages: year from 0 to 6.

States or levels X_i: outcome price of copper in each year as given by the lattice

Return function $g_i(X_i)$ = total benefits at each price level (or state) X_i for a given year (or stage) i.

Cumulative $f_i(K)$ = expected return for a given state given the expected benefits in the subsequent year, taking advantage of the option to abandon the project to maximize profit, plus the benefits from the current year.

Note that dynamic programming problems can be solved forward looking for sequential problems or backward looking when used for options analysis, as in this particular example.