Dynamic Programming Recitation

Dynamic programming (DP) is an optimization 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 – a particular value of the possible states of X_i (used in the 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 stages 1 and 2?
How can I best do __x__ in stages 1, 2, ... and i?

Exercise 7.13 (modified)[1]

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

[1] See different approach to solving a similar problem in file "knapstack.ppt".

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 carried, given the physical constraint of 10 tons. 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. <u>The cumulative return function varies from one problem to the other</u>. 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 the truck, 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 carrying 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 to get 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: $f_3(K) = \max[g_3(X_3) + f_2(K\text{-}X_3)]$

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

$f_3(1) = \max[g_3(1) + f_2(0), g_3(0) + f_2(1)]$
$= \max[\$8 + \$0, \$0 + \$7]$
$= \$8$ (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, $f_i(K) = \max[g_i(X_i) + f_{i-1}(K\text{-}X_i)]$.

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 conceptually to binomial lattice analysis in the copper mine example?

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

Stages: years from 0 to 6.

States or levels X_i: represents all the outcome price values attainable at year i as given by the lattice. For example, X_6 represents all the possible price outcomes in year 6.

Return function $g_i(X_i)$ = total present value cash flow at each price level (or state) X_i for a given year (or stage) i. K here represents a particular price manifestation or value of the state X_i (e.g. price in the uppermost cell in year 5, representing a price sequence up-up-up-up-up).

Cumulative $f_i(K)$ = total present value of cash flows for a given price outcome in year i, which incorporates the present value of cash flows at that state and stage (e.g. uppermost cell at year 5) plus the expected net present value of cash flows in the subsequent stage (e.g. expected net present value cash flow in year 6 if price goes up or down).

In the abandonment case, the cumulative return function incorporates the “real” option to abandon the project if the expected cash flows in subsequent years is lower than the fixed cost of operation if the mine is kept open for an additional year. For each price state K at stage i, two cases need to be evaluated: if the mine stays open and if the mine shuts down. One way to think about the recurrence formula is as follows:

$$f_i(K) = g_i(K) + \max[ENPV(f_{i+1}(X_{i+1}); \text{mine open}), ENPV(f_{i+1}(X_{i+1}); \text{mine close})]$$

In the expansion case, the cumulative return function incorporates the option to expand the project if the expected cash flows in the subsequent year is higher than without expansion. For each price state K at stage i, two cases need to be evaluated: if the production is expanded and if the production stays the same. This suggests the following recurrence formula:

$f_i(K) = g_i(K) + \max[ENPV(f_{i+1}(X_{i+1})$; expanded production), $ENPV(f_{i+1}(X_{i+1})$; same production)]

Note in these recurrence formula X_{i+1} is restricted to two possible states, which is either price up or price down relative to the current price K in stage i. This is analogous to the example above where the state K being evaluated limits the values X_i can take in the cumulative return function. Also, note in this problem that $0 \leq i \leq 6$, and therefore $f_i(K) = 0$ for $i > 6$. Thus for instance $f_6(K) = g_6(K)$.

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