An Introduction to Dynamic Programming

Andrew Hankinson, February 2007

Explanation

Dynamic programming is a method of creating a program, or schedule of events, through which a problem can be broken down into a number of sub-problems (Bellman 2003). These sub-problems may be easier to solve and may also be re-used at other points in the problem solving process, increasing the speed at which the problem can be solved and reducing inefficiencies in the problem solving process.

As a very trivial example, consider the following mathematical problem:

$$40 + 20 + 30$$

Imagine for a moment that we are unable to add numbers larger than 10 without reducing them to their lowest prime number first. How would one solve this problem? We could break it down into a series of smaller problems:

$$(10 \times 4) + (10 \times 2) + (10 \times 3)$$

$$((2 \times 5) \times (2 \times 2)) + ((2 \times 5) \times 2) + ((2 \times 5) \times 3)$$

We immediately see that our computation of 2 x 5 must be done three times - what a waste of time! So we compute 2 x 5 and write the answer in the margin of our page, giving it the variable *y*. We can then substitute our one calculation of 2 x 5 for the value of *y:*

$$(y \times (2 \times 2)) + (y \times 2) + (y \times 3)$$

Our mind finds this formula easier to compute, and we eventually arrive at the answer, 90.

While this situation might be whimsical, it illustrates the main tenet of Dynamic Programming: reduce a complex problem to a series of smaller and smaller problems until we arrive at a problem that we can solve, and build the solution to the larger problem through the solution of smaller problems. As well, the solution for the one sub-problem may also help with other components of the larger problem, so instead of solving the problem many times, we solve the simple problem once and then cache it for later use.

Uses

One possible use of Dynamic Programming is in the field of computer recognition (Amini et al, 1990; Sakoe and Chiba, 1978; McNab et al, 2000). Since computers are 'dumb,' their ability to recognize complex situations is problematic. For example, given a picture of a house, a computer will not be able to recognize that it is a house.

Through dynamic programming, however, the computer can break down the problem into smaller subproblems that it can solve. First, it can determine the boundaries of the various objects, breaking them into discrete geometrical shapes. This can be accomplished by analysing color or shading differences. Once it has determined the gross shape, it can then attempt a finer granularity: does the house have windows, doors, are there trees or people in the scene. Through all of these smaller and smaller steps, it can solve the larger problem and potentially 'understand' the objects in the scene.

In music, this can be important for problems such as optical music recognition, and allowing the computer to 'understand' what is present in a given piece of music based on what it finds in the score. For example, if the computer needs to understand that a given photo-score is in 4/4 time, it can break down the photo into a series of dots and lines, and by breaking this down and then analysing its breakdown it can build an understanding of the score and what it contains, including understanding time signatures and other concepts.

Dynamic Programming is a useful technique for many computer applications and as a general method of problem solving in a many different areas.