Lecture 3 Example: Computational Complexity Analysis

This short description is intended to elaborate on the analysis on complexity discussed in class as part of lecture. The following is the standard formulation of the equipment replacement problem, discussed in lecture 3:

States: age of machine at stage t, s_t

Actions: $a_t \in \{Buy,\ Keep\}$

Rewards:

- $c(s_t) =$ cost of operating a machine of age s_t
- $p =$ price of new machine
- $r(s_t) =$ trade-in value for machine of age s_t

Optimality Equations:

$$v_t(s_t) = \min_{a_t \in \{B,K\}} \{p - r(s_t) + c(0) + v_{t+1}(1),\ c(s_t) + v_{t+1}(s_t + 1)\}, \quad \forall s_t$$

$$v_T(s_T) = -\mathrm{r}(\mathrm{s_t}), \quad \forall s_T$$

The optimality equations can be solved using backwards recursion (sometimes called *backwards induction*). An important question is, how much computational effort is required to solve this problem, and how does it grow as the problem size grows? We can break this down by considering the two major types of computing operations: 1) comparisons within the min{} operation; 2) additions and subtractions. Since we are mainly interested in getting a sense of how the effort grows with problem size we will assume all these operations take approximately the same amount of time (in reality they would vary somewhat depending on the computing platform and software used). Inspecting the optimality equations above yields the following:

At each period, t, and each state s_t, one comparison is made. Thus, if there are n decision stages then the equipment can be 1 year old at the first stage, 2 years old at the second stage, etc.

# of comparisons = 1 + 2 + ⋯ n-1 + n = $\frac{n(n+1)}{2} \sim O(n^2)$

From the optimality equations, at each period, t, and state, s_t , there is 1 subtraction and 3 additions:

# of additions/subtractions = $\frac{4n(n+1)}{2} \sim O(n^2)$

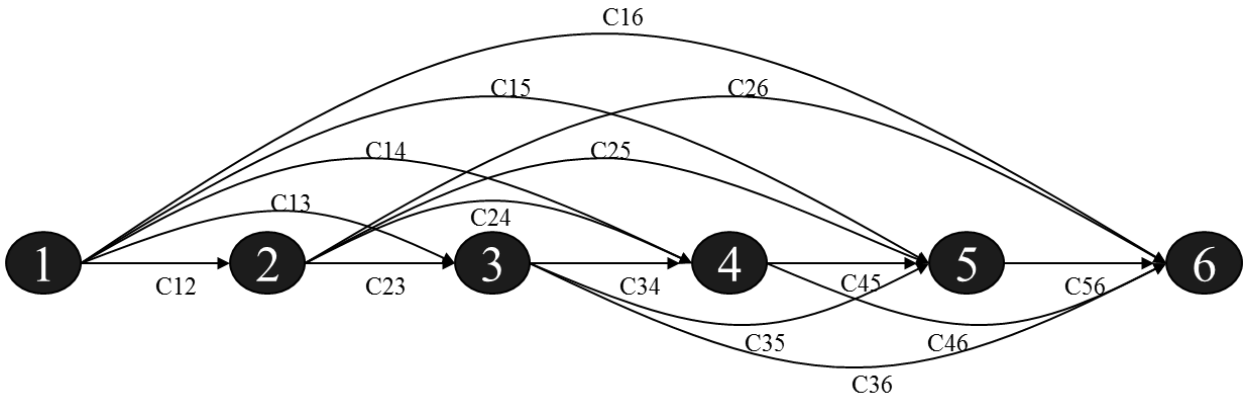
Recall that the notation $O(n^2)$ indicates that there exists some finite number, M, such that the function being approximated is less than or equal to Mn^2 for any choice of n. This provides an approximation of

how the effort changes with respect to problem size. Based on the above analysis, the overall computational effort is:

# of comparisons + # of additions/subtractions = 1 + 2 + ⋯ n-1 + n = $\frac{n(n+1)}{2} + \frac{4n(n+1)}{2} \sim O(n^2)$

and the effort is said to grow quadratically with respect to the size of the problem, as define by the number of decision stages, n.

We also considered a second formulation of the equipment replacement problem as a *shortest path problem* (see slide 11, lecture 3). In this formulation the problem is represented by a directed graph. Following is for the example from lecture 3:



The optimality equations for an n period problem are:

$$v_i = \min_{j=i+1,\ldots,n}\{c_{ij} + v_j\}, \qquad v_n = 0$$

and the computational effort is again in terms of comparisons and additions/subtractions. At each stage i there are n-iI comparisons and therefore the number of comparisons is the same as the prior formulation:

# of comparisons = n+(n-1) + ⋯ 2 + 1 = $\frac{n(n+1)}{2} \sim O(n^2)$.

The number of additions and subtractions is determined by the number of operations to compute the c_{ij}. If we note the definition of these parameters from lecture 3 (slide 12):

c_{ij} = operating cost during years i,i+1,…,j-1 + purchase cost at the beginning of year i - trade-in value at the beginning of year j.

then it follows that the number of additions and subtractions is equal to (j-i +2) and summing over all relevant i, j, for the coefficients we get

# of additions/subtractions = $\sum_{i=1}^{n}\sum_{j=i+1}^{n}(j - i + 2)$

$$= \sum_{i=1}^{n}(1 + 2 + \cdots + n - 2)$$

$$= \sum_{i=1}^{n}(\frac{i(i+1)}{2} - 2)$$

Clearly the first term in the summation is the dominant term. We can use the following formula that gives a close form expression for the sum:

$$\sum_{i=1}^{n} \frac{i(i+1)}{2} = \frac{n(n+1)(2n+1)}{6}$$

to establishes that:

# of additions/subtractions $\sim O(n^3)$ and therefore:

# of comparisons + # of additions/subtractions $\sim O(n^3)$

Thus, this second formulation, based on the shortest path problem, has computational effort that grows more quickly than the original formulation.

Note: as was pointed out in class, the effort for the shortest path formulation of the specific problem given in class is less than estimated above because there is redundancy in the estimation of the coefficients, c_{ij}, since the cost and trade-in estimates are stationary, i.e., they do not vary in time. The above analysis applies to the most general case of the problem in which these parameters may vary in time, but special cases such as the stationary version of the problem may be solved more quickly if there is some special problem structure that can be exploited.