A Dynamic Programming algorithm for the CVCM reconstruction problem

Ahmad Kazemi

The given problem is in fact a multistage optimisation problem. Each day, in which a new machine becomes available to purchase, is a stage. At each stage, we must make decisions regarding reselling the current machine and/or buying a new machine. I decided to employ a Dynamic Programming (DP) approach to solve this problem since:

1. The problem is a multistage problem, and we can decompose the original problem into subproblems at each stage.
2. There are a few possible actions at each stage. Therefore, it is easy to solve the subproblems at each stage.
3. We are limited to have at most one machine at a time, which makes the solution space smaller and tractable for a DP algorithm.
4. There is a natural recursive relation between the stages. Hence, we can write a recursive function, which is a fundamental component in DP algorithms.
5. There are easy "*base subproblems*" for which the optimal solution is obvious regardless of previous stages. Therefore, we can design a DP algorithm that finally reaches to these base subproblems, and then finds the optimal solution of the original problem using the recursive function. The base subproblem for this problem is the reselling decisions at the end of day $D+1$. It is obvious that the optimal solution at this stage is to resell the current machine (if any).

To present the DP algorithm formally, I first define some notations. Set $\mathcal{M}$ denotes the available machines to purchase, where machine $i \in \mathcal{M}$ becomes available in day D_i. There are D days in planning time horizon. We do not need to make a decision in every day. It is enough to make decisions in the days a machine becomes available and only one day after. This is because it is optimal to keep a machine unless we want to buy another machine since the daily profit of all machines is positive. Moreover, we have two types of actions: reselling and buying. Reselling occurs at the start of the day while buying occurs at the end of the day. Therefore, set of stages $\mathcal{T}$ includes two stages for each day in which at least one machine becomes available in addition to the day $D+1$, i.e. $\mathcal{T} = \{(R, D_i), (B, D_i) | \forall i \in \mathcal{M}\} \cup \{(R, D+1)\}$. Each stage $t \in \mathcal{T}$ is a tuple, where its first element shows if it is related to reselling (R) or buying (B) decisions, and the second element is the corresponding day. For the day $D+1$, we need to make reselling decisions only.

Since the *base subproblems* are at the end of time horizon, I decided to develop a *forward* dynamic programming algorithm. The set $\alpha(s_t)$ returns the possible actions at a state (which I will define soon). An action a can be reselling of the current machine or buying new machines and contributes $c(a)$ to the money in hand. For this problem, we can define the state as a tuple of the current machine (m_s) and the available money (v_s), i.e. $s = (m_s, v_s)$. The function $\beta(s_t, a_t)$ determines the next state when action a_t is taken in the state s_t. This function determines the next state based on reselling/buying decisions and the days available in the set of stages. Finally, the recursive function is determined as

$$v_t(s_t) = max_{a \in \alpha(t)} \{c(a) + v_{t+1}(\beta(s_t, a_t))\}.$$

$v_t(s_t)$ shows the money in state s_t in stage t. Moreover, the initial condition to start the DP algorithm is the state with the money equal as the given budget (B) and no machine in hand. The DP algorithm starts with this initial condition and moves forward to reach the final base subproblems that are easy to solve. Then, based on the recursive function, it recursively backtracks from there to build the optimal solution. For especial cases ($M = \emptyset$, $D = 0$, or $B <$ cheapest machine), the solution equals to the given budget (B).