Optimization Engineer at Amadeus: Response to Assessment Task

Parisa Charkhgard

July 13, 2022

Solution Method

The problem's structure makes it a very good candidate for a Dynamic Programming (DP) approach. It can be decomposed into several sub-problems and by optimising each sub-problem, we can optimise the overall problem. The problem description implies that we seek to make a decision on the days that a new machine or a group of new machines are available to be purchased. Should we buy a new machine on those days or not? Since we are only allowed to have one machine at a time and the resell price is fixed throughout the planning horizon, it can be assumed that we sell a machine on a day only if a new machine is purchased on the same day. Consequently, if a day is not the availability day of any machine then it can be skipped and this reduces our search space.

Next, I explain my proposed DP algorithm. For simplicity, let us add two dummy machines O_1 and O_2 to the problem with purchase price, resell price and daily profit of zero. Dummy machines O_1 is available on day 0 and O_2 is available on day $D+1$. According to the problem assumptions, we should purchase these two machines on their available days. Table 1 describes parameters of the problem.

Table 1: Problem parameters

$\mathcal{H} = \{1, \ldots, D\}$	The set of days in the planning horizon
$\mathcal{M}$	The set of machines
D_m	The day on which the machine m is available for sale
P_m	Price for machine m
R_m	Resell price for machine m
G_m	The daily profit generated by operating machine m
C	Initial money available at the beginning of the planning horizon
$\mathcal{N}_d^+$	The set of machines that become available on day d., $\mathcal{N}_d^+ = \{m \in \mathcal{M} \colon D_m = d\}$
$\mathcal{N}_d^-$	The set of machines that become available before day d., $\mathcal{N}_d^- = \{m \in \mathcal{M} \colon D_m < d\}$
$\mathcal{T}$	The set of decision days.,$\mathcal{T} = \{d \in \mathcal{H} \colon \lvert\mathcal{N}_d^+\rvert \geq 1\} \cup \{0, D+1\}$

Let $V_d(m)$ denote the maximum (optimal) profit from the start of the planning horizon until the end of day d where machine $m \in \mathcal{N}_d^+$ is bought. By this definition, the initial state is $V_0(O_1) = C$. This means that at the end of day 0, machine O_1 is owned and the profit is C. If m is a machine with available day of d, then for calculating $V_d(m)$ we need to find the best strategy to sell a previously purchased machine and buy machine m on day d. We should basically look at all possible machines that could be available to us before day d and calculate the best/most profitable path. Consequently, the recursive equation would be:

$$V_d(m) = f\left(\max_{m' \in \mathcal{N}_d^-}\left(V_{D_{m'}}(m') + (d - D_{m'} - 1) \times G_{m'} + R_{m'}\right) - Pm\right)$$

where $d \in \mathcal{T}\backslash\{0\}$, $m \in \mathcal{N}_d^+$ and

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ -\infty & \text{otherwise.} \end{cases}$$

$f(x)$ is added to make sure that purchasing a machine while there is not enough money is never in the optimal solution. On day $D+1$, CVCM will sell any machine that it owns (and purchase dummy machine O_2), then the total profit would be $V_{D+1}(O_2)$. The value of $V_{D+1}(O_2)$ is basically what we are looking after and it can be calculated through:

$$V_{D+1}(O_2) = \max_{m' \in \mathcal{N}_{D+1}^-}\left(V_{D_{m'}}(m') + (D - D_{m'}) \times G_{m'} + R_{m'}\right)$$

I used a top-down approach and implemented the algorithm in Python. The outcome for the provided instances can be seen in Figure 1. Figure 2 shows the result for a large instance with 600 machines and 1000 days. You can find this instance with the name `large_instance.txt` in the repository. As can be seen, the algorithm works well even for this large instance.

```
(base) parisa@dell:~/Desktop/task_amadeus$ python main.py instance.txt
Successfully received 6 instance(s).
Solving Case 1.
Solving Case 2.
Solving Case 3.
Solving Case 4.
Solving Case 5.
Solving Case 6.
All instances are solved. Printing the results.

Average run_time: 7.39e-05 seconds

Instance input  Instance output
6 10 20         Case 1:  44
0 11 30         Case 2:  11
1 12 30         Case 3:  12
1 10 2          Case 4:  10
2 10 11         Case 5:  39
2 10 11         Case 6:  39
```

Figure 1: The result of the 6 given instances

Remarks

- In special cases with $D \leq 1$ or $\mathcal{M} = \emptyset$ or $C < \min_{m \in \mathcal{M}\backslash\{O_1,O_2\}}\left(P_m\right)$, the solution equals to the initial money available at the beginning of the planning horizon, i.e., $V_{D+1}(O_2) = C$.

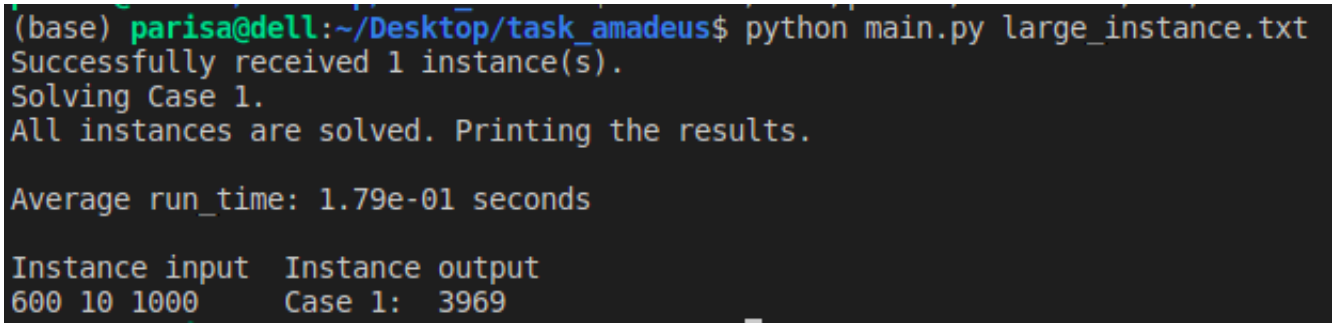

Figure 2: The result of a large instance with 600 machines and 1000 days (solved in less than 0.2 seconds)

- We store the results for each sub-problem ($V_d(m)$) to avoid repetitive computations (memoization).
- I used a top-down approach to solve the problem. Another similar approach would be a bottom-up approach where you start from day 0 and go forward in time and make a decision on each decision day.
- It is possible to create a graph representing the problem. Node (m, d) in this graph would be the state of owning machine m at the end of day $d = D_m$. There will be an arc from $N = (m, d)$ to $N' = (m', d')$ with the weight of $w_{(N,N')} = (d' - d - 1) \times G_m + R_m - Pm$ if $d < d'$ (i.e., $D_m < D_{m'}$). $w_{(N,N')}$ represents the contribution of buying machine m in day d, using it until day $d' - 1$, and selling it on day d'. The problem is to find the longest path from $(O_1, 0)$ to $(O_2, D + 1)$. We can remove the arcs with negative weights since they won't be in an optimal path.