The CVCM reconstruction problem

Asghar Moeini

The problem has a very specific structure which makes dynamic programming a powerful solution method. In particular, the state on each day is the machine owned at the end of the day an associated amount of money made by the end of the day. Now we can simply go forward through time and work out on each day, and for each possible machine, what would be the maximum amount of money we can have if we are able to purchase that machine (looking at all possible machines that we might have had on day *d*-1 with their associated cash balances). If there are one or more ways to purchase machine *m* on day *d*, pick the one that gives the most money and store it as the current state. At the end work out the highest total value (residual + cash) and work backwards.

Note that, on each day we only consider those machines that are available on that day and days with no new machine available can be skipped entirely. It should be mentioned that machines that haven't yet turned a profit are also not worth considering for replacement. This can decrease the search space and therefore makes the algorithm more efficient. However, I don't think any of this significantly affects the worst-case complexity of the approach which is definitely polynomial even with a simple implementation. I imagine in practice very large instances could be solved efficiently. Following simple MATLAB code is the implementation of the designed algorithm.

```
clear all, clc
% Please enter the input below
N = 6; C = 10; D = 20;
M = [6 12 1 3; 1 9 1 2; 3 2 1 2; 8 20 5 4; 4 11 7 4; 2 10 9 1];
%%%%%%%%%
if N == 0 || C == 0
    OptimalProfit = C
else
    [Ms Mr] = sort(M(:,1)); % Relabeling the machines based on their purchase days
    M = M(Mr,:);
    M = [M; [ D + 1 0 0 0]];  % Here we add one imaginary machine in day D + 1
    d = M(:,1); p = M(:,2); r = M(:,3); g = M(:,4);
    e = 0.0001;
    for i = 2:N
        if d(i) - d(i-1) <= e
           d(i) = d(i-1) + e;
        end
    end
    n = length(d);
    a = zeros(1,n);
    a(1) = C-p(1);
    k = zeros(n,N);
    skipNum = [];
    for i = 2:n
        for j = 1:i-1
            if ~ ismember(j,skipNum) % This removes the machines that we
                                     % cannot buy because of lack of money
                k(i,j) = a(j) + (d(i) - 1 - d(j)) * g(j) + r(j);
                % k(i,j) determines the maximum money on day d_i if M_j is
                % resold on that day
            end
        end
        z = max([k(i,:) C]) - p(i);
```

```
        if z >= 0
            a(i) = z;
        else
            skipNum = [skipNum, i];
        end
    end
    OptimalProfit = round(max(a))
    % OptimalProfit is the maximum money that we can have in day D+1
    machine = [];
    ss = n;
    for i=1:n
        [rr ss] = max(k(ss,:));
        if rr <= C
            break
        else
            machine = [ss machine];
        end
    end
    if isempty(machine)
        fprintf("We do not need to buy any machine")
    else
        machine  % Solution: this determines which machines we need to buy
                 % Note that machines are relabelled based on their
                 % availability for purchase
    end
end
```

Results on the test problems:

Example Input	Example 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
0 0 0	Case 7: 0