Report: Assignment 3, Stockmarket

L. Schouten s2660148
D. Macquine s2592991

November 6, 2021

1 Introduction

For this assignment it was necessary to write an algorithm to determine the highest amount of budget that could be achieved at the last day. For the implementation of the algorithm, the dynamic programming approach is used. The dynamic programming approach makes it possible to yield the optimal solution without the need of having to enumerate all possible sequences of portfolios. Adding to that, another algorithm was written that makes it possible to retrieve the sequence of portfolios over the days which would result in the highest amount of budget on the last day. This algorithm uses a backtracing approach to retrieve the sequence of portfolios resulting in the optimal solution.

2 Recursive formulation

Figure 1 shows the recursive formulation of the dynamic programming algorithm.

$$M[i][j] = \begin{cases} \max\{-1, b + L(j,0,i)\}, & if\ j = 0 \\ \max\{-1, \max_{0 \le k \le \max(i)} \{(M[k][j-1] * r + L(j,k,0)) + L(j,0,i)\}\}, & if\ j > 0 \end{cases}$$

Figure 1: Recursive formulation

To calculate M[i][j], which is the maximum budget which can be achieved with portfolio i on day j, two cases exist.
In the first case the day is equal to 0. This means that it is the first day. The maximum budget for that day is the maximum of the starting budget (b) + the transaction cost (L) or -1. The -1 value represents a portfolio that is not reachable with the starting budget. The transaction cost are determined with the function L which is a function with three parameters where the first argument 'j' represents the day, the second argument represents the current portfolio that the user possesses, the third and last argument is a portfolio that is aimed to be possessed. When the value of the bought portfolio is higher than the value of the starting portfolio on day j, the value returning from L will be negative. When the value of the bought portfolio is lower than the value of the starting portfolio, the value returning from L will be positive.

The second case is where the day is not zero, so this is the case for all other days than the first day. Here the calculation for M[i][j] is a bit different. To calculate M[i][j] it is necessary to loop over all the portfolios of the day before (j-1) to request yesterdays budget for the particular portfolio. Ranging over all budgets for all yesterdays portfolios can be done with M[k][j-1], where

k is a number in the range from 0 to the number of possible permutations of portfolios that exist. Yesterdays budget, today, has become worth more due to the interest factor and thus the value of yesterdays budget is multiplied with interest factor 'r'. To find out what portfolio k is worth today, the portfolio will be liquidated by using L(j, k, 0) where j is again the current day, k is the index of one of yesterdays portfolios and 0 is the empty portfolio having no stocks. Thus this algorithm calculates the value of having no stocks, since it sells the whole portfolio to get the the empty portfolio. The budget of portfolio k from yesterday multiplied with the interest factor and combined with the money from the liquidated portfolio denotes the amount of money which can be spend today on portfolio i, this amount is called the liquidized equity. To calculate the transaction cost of buying portfolio i from the empty portfolio L(j, 0, i) is used. The remaining budget after buying portfolio i can be calculated by taking the transaction costs from the liquidized equity. the remaining value may either be -1 if M[i][j] can not be bought or some remaining budget. After having iterated over all (yesterdays) portfolios k and having calculated all k's liquidized equity, the k which yields the highest liquidized equity is used to buy i. After buying i, the remaining budget is stored on M[i][j]. If it is impossible to buy portfolio i because there is not enough budget to buy the portfolio, the value -1 is stored instead of the remaining budget.

3 Optimal strategy

After the memory table is filled bottom up, it is possible to derive the strategy which leads to the highest budget at the end of the term. To find out what strategy results in the maximum budget, backtracing can be used. To find out what the best strategy is, the maximum budget on the last day should be taken and all portfolios of the day before should be enumerated. This is done to try out which portfolio of the day before, together with the free budget of the day before, results in this maximum budget. When the portfolio that results in the maximum budget is found, the same steps must be repeated for maximum budget of the day before and so on, until the maximum budget of day one and the portfolio of day zero (the first day), that results in this maximum budget, is found. When the portfolios resulting in the highest budget for some day are stored in between, the optimal strategy of portfolios is retrieved.

4 Time complexity & space complexity

4.1 Time complexity

The time complexity for this method is as follows. For the first day, for all the possible portfolios, the budget must be calculated. Portfolios are represented by bitstrings where a 1 means that a stock is present in the portfolio and a 0 means that the stock is not present in the portfolio. Having n stocks, there are 2^n permutations of portfolios. Therefore, to calculate the budgets for the first day, 2^n operations are necessary. For the other days, it is necessary to find out which of the portfolios the day before would give the highest budget for today. The highest budget of yesterday can be used for all the portfolios today. This results in a time complexity of all the other days combined of $(t_e - 1) * 2^n * 2$. The time complexity of this method in terms of Θ-notation therefore is: $(t_e - 1) * 2^n * 2 + 2^n$.

4.2 Space complexity

The method uses a 2 dimensional array to store values. In this array, each column represents a day and each row represents (the index of) a possible portfolio. The array should be able to

contain all maximum remaining budgets for the different permutations of portfolio's for each day. This means that the numpy array should have 2^n rows. Furthermore, there are t_e days for which the remaining budget should be stored. Therefore, the space complexity of this method described with the Θ-notation is $2^n * t_e$

4.3 Improved Space complexity

It is possible to improve the space complexity. One way to improve the space complexity is by using only two columns to store information instead of t_e columns to store information. The time complexity of this method in Θ-notation would would be $2^n * 2$. The reason that only two columns are necessary for storing information is that the recursive formulation in case of the first day only uses the starting budget and the transaction cost of creating a certain portfolio so here just one column would fulfill. To be able to calculate the information for the other days, two columns are necessary, one for storing the information for a certain day and the other because the recursive formulation uses the information of the day before. For all other days than the first day, when the second column is filled based on the information of the day before (the column before), the first column may be wiped and the information of the second column may be transferred to the first column. Hereafter, the second column may again be used to store information. Therefore just two columns would fulfill. The consequences of only using two columns instead of one columns is that it will no longer be possible to find out the path to the optimal solution by using backtracing, because the necessary information about which sequence of moves led to the optimal solution no longer exists.

5 Summary

This report is about the implementation of dynamic programming and backtracing. The dynamic programming approach is used to find an optimal allocation of resources of liquid cash and stocks with the aim of ending with the highest possible equity. The backtracing approach is used to find one way to the optimal solution. The dynamic programming approach and the backtracing approach work as expected in advance.

As mentioned above, it would be possible to improve the space complexity of the algorithm at the cost of not being able to traceback a path to the optimal solution anymore. Furthermore, when the interest gains and investing in a stock have the same result, the algorithm will choose not to buy the stock and keep the cash at hand to gain interest. The algorithm may be changed to prefer investing in stocks above getting interest if both yield the same result depending on user demands.
Furthermore, when different paths to the optimal solution exist, the traceback algorithm will return the path first to encounter, and not both. This is because the algorithm builds up the path gradually from day to day. When a path from the previous day to the desired budget is found, the algorithm stops searching for other paths from the previous day to the budget. The algorithm could be altered to find all the paths that exist if it is decided that one path might be preferable over another. Altering the algorithm to find all the paths will however come at the cost of speed because it can't move on to the next day when a path from one day to the budget of the next day is found.