Programming Tools and Technologies for Data Science

Apostolos Kakampakos (03400133)
Data Science & Machine Learning
apostolos.kakampakos@gmail.com

February 27, 2022

National Technical
University of Athens

Structure of the Assignment

This work is composed of this pdf file explaining the work, two txt files named small.txt and large,txt, as well as 3 python files named:

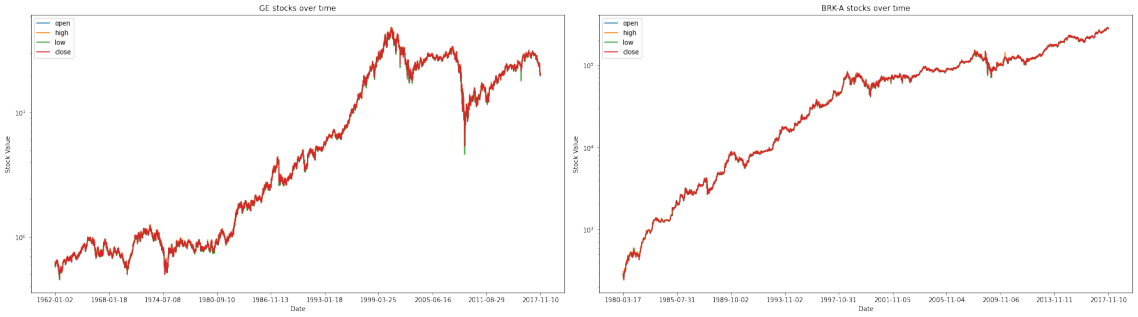
- lib.py
- simulation.py
- generate.py

The first file, lib.py contains all classes and functions useful for running all programs. The second file, simulation.py is responsible for simulating the trade transactions of a given txt file and producing graphs and general information. The third file, generate.py, generates a txt file containing commands that are made using the algorithm I implimented for this work. Both small.txt and large.txt were made by running this python file.

Finally we have the Stocks folder (which i didn't send) containing all company stocks.

Hierarchical viral profits

In this section I will explain the main strategy implemented for this assignment. I called it Hierarchical viral profits, and will explain why. Viral denotes the fact that we try to "infect" as many different companies as possible. Note that we only picked a special subset of exponentially growing companies. Therefore, the hope is that as time goes on, we will gain hold of as many stocks of all growing companies, giving a high reward in the future. Hierarchical denotes the fact that when we buy more stocks of an already "infected" company, we always prefer the one with the highest return deep in the future. This way as we go forward in time we buy stocks from all companies in our pool, then they all slowly converge to the company with the biggest value.



(a) GE stocks (b) BRK-A stocks

These two figures are examples of companies contained in the selected subset of all companies. More specifically I picked GE stocks because they are one of the earliest and cheapest stocks as a seed for the algorithm. BRK-A stocks are the most valueable ones in our dataset, with wild fluctuations, which are most valuable for intra-day trading.

As we will see, the main money-making part of the algorithm happens abruptly and exponentially as we go near the 2000's. This happens both because all companies grow in presence and value.

Crucial, for the algorithm is to pick only growing over time companies, that are also exploding in value.

Classes of lib.py

This section will explain all the classes contained in lib.py

lib.wallet

One of the main classes is called wallet. This class is responsible for executing all transactions and calculations. Furthermore it keeps track of the balance as well as the portofolio of stocks. Those are important both for calculations as well as for plots we will see later. The wallet class is used in simulation.py as well as in the agent class defined later.

lib.company_stocks

The main class dealing with reading company stocks is called company_stocks. This class contains all stocks of a specific company from beginning to end. This class is used mainly in the class defined below.

lib.stocks_dataset

This class is responsible for filtering and holding all companies deemed successful enough to be used for trading. Its main functionality is receiving a day and returning a dictionary with company names as keys and values a tuple containing all daily relevant information of that company.

The filtering functionality is performed from the method keep_good_companies which has 2 parameters. Margin and slope.

- margin: Only select companies with future high stocks bigger than margin
- slope: Fits a line for stocks over time in logarithmic scale and only picks slope bigger than the given parameter.

Both parameters are used to select only the best companies in our dataset. For example for small.txt we picked slope = 4.0 and margin = 300. This resulted in selecting only 10 companies. Moreover, for large.txt we picked slope = 4.0 and margin = 100, selecting only 40 companies. Note, when we are performing filtering, some warnings are printed, which are for misbehaving data values of companies which are simply discarded.

This class is mainly used in the agent class, defined below.

lib.agent

This class performs the algorithm defined in the previous section. It contains the main methods act and update wallet. Method act selects between policies implementing our strategy and update wallet, simply updates the wallet of the agent (defined above) in the ending of transactions of a day.

Now we will describe all policies the agent makes during runtime.

Only one policy is made per day. The algorithm selects them in the order presented next and only selects one if the previous cannot be made, with the latter being dismissed. If no policy can be made (when a day simply has no values or otherwise) no action is performed.

sell_all_policy

This policy is used only when the day is reached, which we define setting final_day parameter to a specified value. This policy simply attempts to sell all stocks the agent owns until they have none, while respecting the spacetime safety concerns (less than 10% of volume of transactions). For the small.txt this policy laste only a day, while for the large.txt, since it accumulated way more than the volumes of daily transactions of many companies, therefore it takes a while to sell them all because of the spacetime safety concerns.

sell_policy

This policy is used for 2 reasons. First, for setting a day for selling for any company owned that doesn't have a sell date, when that company has more than one stock. Second for selling when the sell day comes all but one stock (to keep the infection going). This also must respect the spacetime safety limitations. The setting of memory is made using the premonition method.

infection_policy

This policy is implemented to buy stocks from a new company. I called this infection, since the whole point is to find the cheapest low stock of a new company and buy it to infect it. Then other policies will take care of the company.

buy_policy

This policy is used to buy as many stocks as possible, of each company already infected. Here we also implement the hierarchical part of the strategy, since we prefer buying stocks having bigger future stock value, than simply choosing the ones that are cheapest now. This means that as the algorithm progresses we will drive all stock possesions to the company with the future highest stock value.

intra_day_policy

This policy is the main money-making action of our strategy. It is defined by a margin. When the policy is activated we search for each company's stocks in a day and find if selling in the open and buying when its low is bigger or lower than selling when high and buying when its closing time. Then we find if the profit from doing the best of the above actions for one stock is bigger than a specified margin (0.1 for the large.txt and 0.5 for the small.txt) then if it is we simply sell and buy as many as we can, while we respect spacetime safety restrictions.

no policy

When none of the requirements are met for any policy, no action is performed. This happens when no money can be spend on buying or the margin for intra day trading is simply not surpassed.

small.txt

This section presents the results of producing the small.txt file using generate.py. This was implemented using first slope = 4.0 and margin = 300 when selecting our data using the lib.stocks_dataset class. Furthermore we chose a margin of 0.5 for intra day transactions and the sell day as 1999-05-16.

The profit generated at the end is 395,903.92...

First, the graph showing how both balance and portofolio values evolved over time.

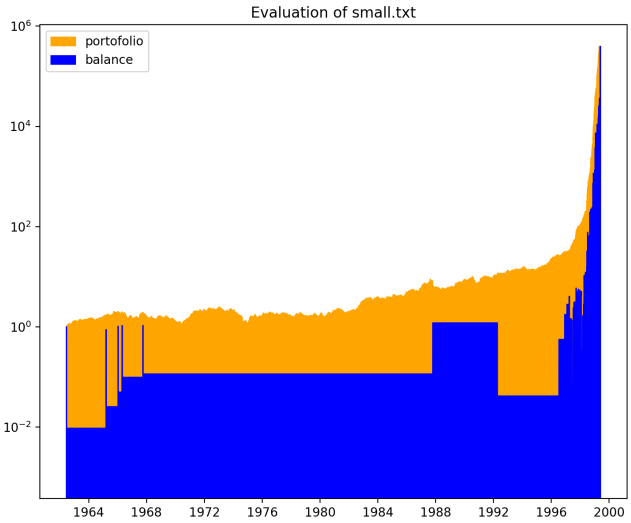


Figure 2: Balace and portofolio of small.txt

As we can see for first decades both balance and portofolio have small values and then for the last 4 years they explode! This is because of how intra day trading picks up steam and a multitude of moves are performed pretty much every day.

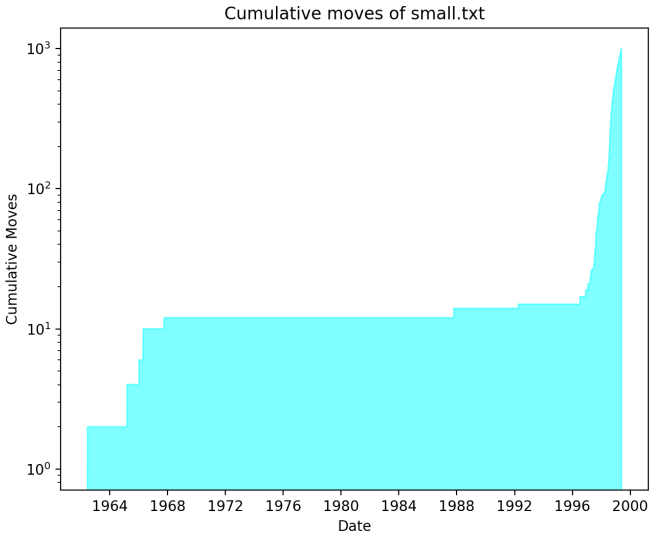


Figure 3: Cumulative moves of small.txt

This can be seen by the cumulative moves graph produced above. As we can see move count jumped in the last 4 years, while the algorithm started from the 60's.

Finally, another graph of importance is to see the total number of stocks owned over time.

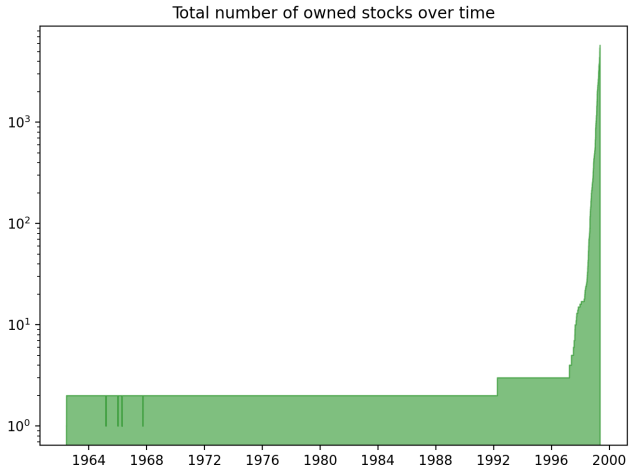


Figure 4: Stocks owned over time of small.txt

As we can see this method picks up after many years but explodes exponentially. I had to put the sell day in the 99's so as to simply keep the moves under 1000.

large.txt

Selecting our data with slope 4.0 and margin 100.0 as well as having an intra-day margin 0.1. Also, the sell-date now is 2001-01-25.

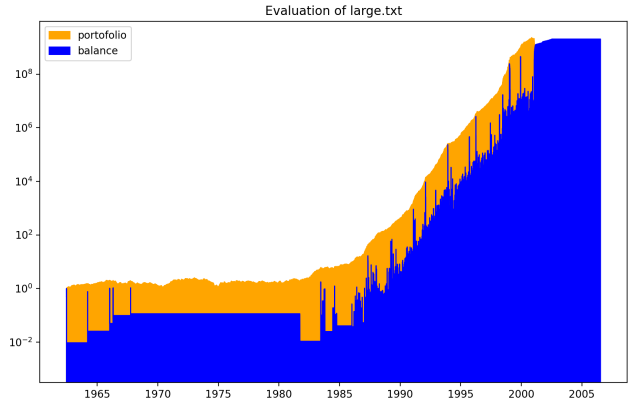


Figure 5: Balance and portofolio of large.txt

The profits produced are about 2,034,405,408 dollars. Total moves are 56522.

Figure 6 shows how the exponential growth started in the 80's and reached the 2 billion mark after 20 years. This program took more than 3 hours to run, with most of the time processing the few late years. One could set the sell date even later, but simply the amount of time it takes is too much to justify, since a good result was already produced.

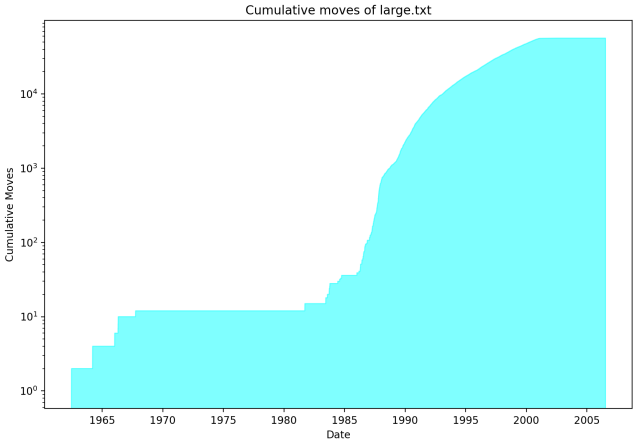


Figure 6: Cumulative moves of large.txt

The above figure simply represents how rapidly moves grew over a decade, while being pretty much stable for previous decades.

Furthermore, let us see how total stocks owned over time!

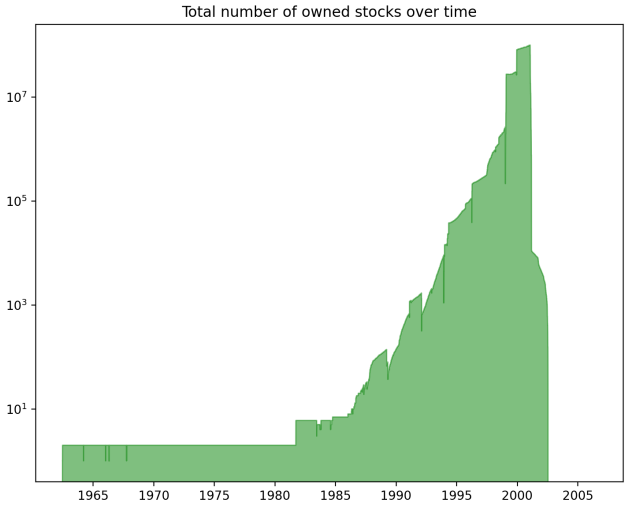


Figure 7: Stocks owned over time of large.txt

Figure 7 shows the rapid growth of moves as well as the decline after the sell-day has triggered the sell-all policy. Having many more stocks of the first company (BRK-A) than its transaction volumes, it actually took a year to sell all stocks, while respecting spacetime safety restrictions!!!

Discussion

This algorithm was thought as a way to go wide instead of tall but slowly evolved as a mixture of the two. Obviously, nobody can claim that it is optimal or that it has no further room for optimization. But, still it has a unique behaviour, since for the most part it is pretty much stable, then it exponentially grows providing many profits.

Note, 0.1% of 2bn is already 2 million, which is certainly a good amount if I ever am able to timetravel.