**REPORT**

**APPROACH**

The problem statement given was to maximize a linear equation with 5000 decision variables and 500 constraints. The limit of the community version of CPLEX is 1000. Hence its impossible to solve the problem as a whole. This is the reason why sifting procedure is used.

The first part of the code involves processing the input data from the file. The file has been fed without any modification and the A,b,c matrices have been extracted. It makes sense to redefine the problem statement here for ease of understanding:

Problem: Maximize , Subject to -> 1

Corresponding dual: Minimize , Subject to -> 2

Now I first spent some time installing and exploring CPLEX and its python API. Its support with numpy is one reason why I choose to do this in python. The function solve\_primal\_and\_dual takes the A,b,c matrices and solves it and its dual to give us xw\* and pi\* which are the solutions for the problem and its dual.

The sifting process mentioned in the research paper was for a different problem. For this one, the process would look like:

* Take some initial set of columns that give a feasible solution. This part was easy as all prefixes upto 1000 gave a feasible solution. I choose the first 500 columns as the initial set of W.
* Now we construct Aw and Cw and then find the initial pi\*.
* Now we need to ultimately satisfy the condition of our dual problem. Hence pi\* must satisfy.
* Now we take all those columns that do not satisfy this condition and add them to a list rho.
* I set t = 100, meaning we can only take 100 columns at once. Hence I sported rho and took the first 100 ones every time. Repeat this process until the set size converges which means there aren’t any more negative columns left.

After iteration 4 itself, the set size started converging and it finally converged to 896. This means we managed to reduce 5000 columns to 896. In other words, our model choose to maximize this rather than the whole set of columns as they turned out to be good candidates.

Finally, I choose the xw\* which gave this solution and found the corresponding X\* by setting all those variables that weren’t present in W to be 0.

The final objective value that I was able to obtain was 1515714.6139719822

After this, I checked the solution using a short code to make sure all constraints were satisfied and they were.

**IMPLEMENTATION - SPECIFICS**

In this implementation, the normal matrix multiplication was used instead of fast vectorization (Section 2.1) as the process was already fast enough.

As for pricing algorithms (Section 2.2), I used the lambda pricing algorithm as the research paper clearly mentions that it gives a faster convergence rate.

As for the choice of “t” (Section 2.3), different t values gave different iterations. A bigger value of t gave less iterations but a lot more columns considered in the end while a smaller “t” gave more iterations but lesser columns considered in the end. T= 100 seemed like a middle value. It is important to mention that none of these values of t changed the objective function value.

For solving the problems to find xw\* and pi\* (Section 3), CPLEX was used along with the docplex model. It necessary to run the code on a system with CPLEX support and docplex package installed for the code to work.

The final solution of all decision variables has been included in the output file.