Overview

The original objective of this project was to optimize a financial portfolio to minimize risk while adhering to various strictures, which started out as very vague. With this much it was suggested that the simplex algorithm could be used to solve such a problem if it was cast as a linear program. With this in mind I relearned linear programming using the CLRS algorithms text.

It was only after that that I addressed the details of the problem to be solved, so do nothing more than solve a maximization problem on a set of financial holdings against a set of linear constraints. Some research on this suggests that a real problem of this nature becomes more complicated and requires something more along the lines of a quadratic programming problem (I don't claim to be an expert on this) so what I came up with may not be entirely relevant to the financial industry.

Newly stated the problem this: You have a set of n instruments, which you can buy into a portfolio with a set of constraints that we will call a model. The constraints are based on properties of the instrument and will have to be linear in nature, so if I have a property P represented by a floating point number p_i for instrument i in (1 … n) then:

$$\sum p_i x_i \leq b_i$$

for some set of numbers b_i which we will call constraints.

The first property to consider is price and we will assume that there is a fixed limit on total cost. So where p_i = Price, b_i is amount available to buy the securities.

Assume we also have a property called Risk, which is some factor that can be applied to the holdings weighted by value so that the total is some amount that is considered at risk with some degree of probability. The value of b for this is again an upper bound.

To this add some entirely fictional property that represents exposure to some industry say ExpRE that you want to keep to some maximum.

Another property will be expected return on investment; I will call it RoR, taking the risk of someone thinking I think I know what I am talking about. The weighted sum here represents the amount you expect to make over some period and thus is a practical candidate for the objective function. That is to say if $c_i = p_i$ for all i then we wish to maximize:

$$\sum c_i x_i$$

Planning

To solve this I used the Simplex algorithm. This involves laying out the variable coefficients (p_i) in an mxn matrix and keeping the constraints (b_i) in an m vector and the objective coefficients (c_i) in an n vector. You then create a "slack" format for the matrix, which adds 3 more variables under further constraints (that they must be greater than zero). This allows one to manipulate the constraints as a set of equations where the new variables have their own coefficients.

My first goal was to write some matrix operations and show that as progress. Most of these proved of no value but some surprisingly were useful. transpose was the most useful. It was very difficult to know what operations were necessary until the simplex algorithm was underway.

Following that I attempted to implement some form of the algorithm using the mostly imperative formulation in the text. At first this proved surprisingly easy for the initial test however it soon came apart and my TF suggested I really should use a functional approach.

I ended up expanding the original matrix to a m+n square matrix, since I had to maintain the coefficients for the m additional slack (or basic) variables. This also required maintaining list of basic and non-basic (or original) variables to keep track of which were active at any time, since the algorithm involves moving them from one to the other at each step (moving from one to the other is called a pivot).

I was able to adopt a more functional approach by creating functions around these lists. These functions would take a matrix or vector and if the rows and extract the appropriate items for a particular operation.

Design and implementation

The input would look like this:

```
Instrument: MSFT
-RoR 3.5
-Risk 1
-ExpRE 2
-Price 25

Instrument: AGE
-RoR 4.1
-Risk 1
-ExpRE 2
-Price 40

Instrument: AAP
-RoR 12.0
-Risk 3
-ExpRE 5
-Price 200

Objective: >RoR

Constraints: Price<50000 Risk<10000 ExpRE<30000
```

From this you would create a matrix:

25.	40.	200
4.	4.	2.
2	2	5

and two vectors:

the constraint vector b: (50000, 10000, 30000)
and the objective vector c: (3.5, 4.1, 12.0)

You would maximize:

$3.5x_1 + 4.1x_2 + 200x_3$

while maintaining:

$50000 \geq 25\ x_1 + 40\ x_2 + 200\ x_3$
$10000 \geq 4\ x_1 + 4\ x_2 + 2\ x_3$
$30000 \geq 2\ x_1 + 2\ x_2 + 5\ x_3$

To solve this I create a larger matrix:
To solve this I create a larger matrix:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
25.	40.	200	0	0	0
4.	4.	2.	0	0	0
2	2	5	0	0	0

With lists basic = 4,5,6 and non-basic = 1,2,3

This represents a system of equations:

$z = v + 3.1\ x_1 + 4.1\ x_2 + 12.\ x_3$

$x_4 = 50000 - 25x_1 - 40\ x_2 - 200x_3$
$x_5 = 10000 - 4x_1 - 4\ x_2 - 2x_3$
$x_6 = 30000 - 2x_1 - 2\ x_2 - 5x_3$

Where we want to maximize z and v is a "slack" variable that is used by the algorithm. By a series of algebraic manipulations we maneuver the last 3 variables into the main matrix while also manipulating the vectors b and c. By doing this until the appropriate set of conditions is met we eventually arrive at a vector c that shows that a solution has been arrived at.

Reflection

This was torture, like all the assignments. Everything that worked was a pleasant surprise as it always with Ocaml but these always seem to come in big thunks. Once it works it seems to work very well.

I learned late in the project that you have to write test programs as you write the functions. Unfortunately it is hard to predict the structure the program will take if the algorithm you are following is entirely in an imperative format.

One of the hardest things was knowing how the builds worked. It is perhaps unfortunate that I didn't pay more attention to the makefiles for the assignments because whenever I got an unbounded variable that was in an open'ed module it took too long to solve it. Eventually I just followed the makefile from ps7.

I would not have used any other language than Ocaml because I took the course to learn functional programming and that is the only functional language I know.