DSA/ISE 5113
Advanced Analytics and Metaheuristics
Homework #4

March 5th, 2023

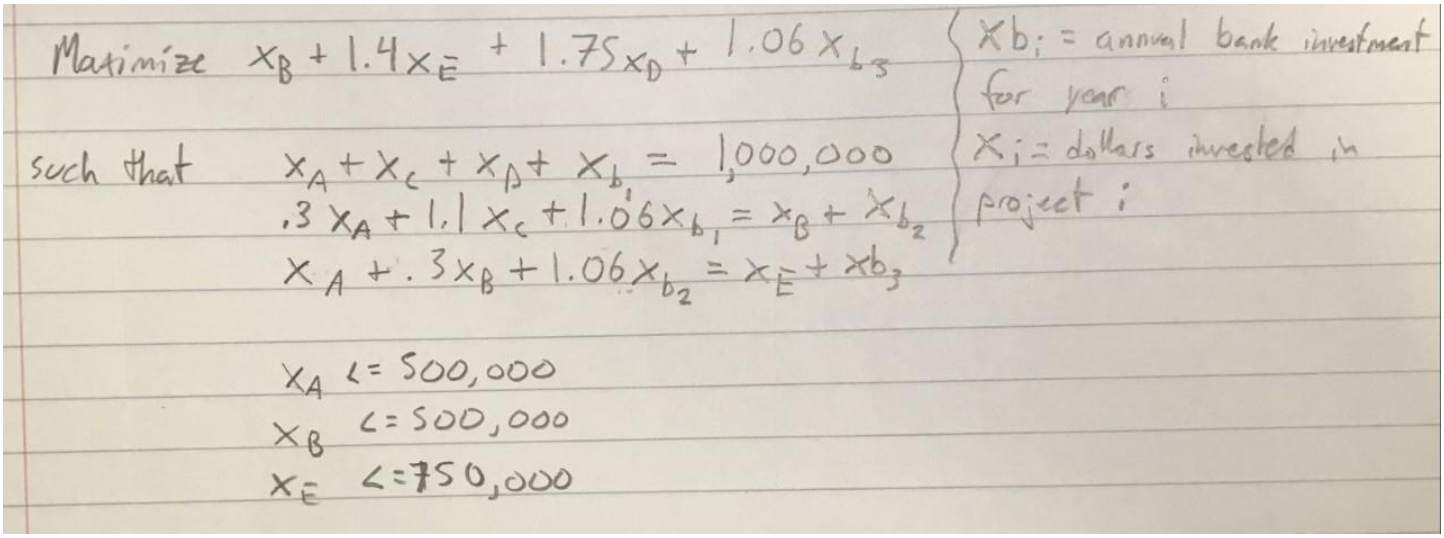
Adam Lambert

Marshall Baldwin

Question 1: Titan Enterprises Case Study

(a) Formulate and solve Titan's investment decision problem as a linear program (use AMPL).

i) Maximize Return only.



The above model was implemented in AMPL as follows:

```
#clear memory
reset;

#choose solver;
option solver cplex;

#create sets to define variables
set Years = {1,2,3}; #years for bank investments
set ProjectIDs = {1,2,3,4,5}; #ID corresponding to each project: 1=A, 2=B,..., 5=E

#initialize model parameters
param initInv; #initial amount to be invested in dollars
param riskCoeff {p in ProjectIDs}; #risk per dollar for each project (for part iv)

#Set problem variables
var x {i in ProjectIDs} >= 0; #dollars invested in project i
var xb {i in Years} >= 0; #dollars invested in bank for year i

#the following are specified by Fig. 1 in the homework PDF

#set objective function
maximize investment_return: x[2] + 1.4*x[5] + 1.75*x[4] + 1.06*xb[3];

#specify constraints
s.t. c1: x[1] + x[3] + x[4] + xb[1] = initInv;
s.t. c2: .3*x[1] + 1.1*x[3] + 1.06 * xb[1] = x[2] + xb[2];
s.t. c3: x[1] + .3*x[2] + 1.06*xb[2] = x[5] + xb[3];

#add project investment limits (see HW2)
s.t. limA: x[1] <= 500000;
s.t. limB: x[2] <= 500000;
s.t. limE: x[5] <= 750000;
```

Note that the commands to solve and display results have been cropped out. The full code is included in the zipped homework files.

Running the model file results in:

```
Solving the model we can see that the following investment return can be made:
investment_return = 1797600

This is done using the following investments:
x [*] :=
1   5e+05
2       0
3       0
4   5e+05
5  659000
;

xb [*] :=
1       0
2  150000
3       0
;

Note that the risk in this scenario is:
risk = 182950
```

This returns the same results from HW2, a total return of $1,787,600. Note that the calculation of risk will be explained in part (ii). The value of 182950 for risk will be used later in our epsilon constraint method.

(ii) Minimize risk only.

The model was slightly tweaked by changing the objective function to minimize risk rather than maximize return. We assume that a "risk score" can be calculated by multiplying the dollars invested in a project by that project's coefficient of risk. This is unweighted by monetary return. We also assume that the annual investments in the bank have zero risk and can therefore be neglected from our risk calculation. Our model and data file are as follows:

```
#create sets to define variables
set Years = {1,2,3}; #years for bank investments
set ProjectIDs = {1,2,3,4,5}; #ID corresponding to each project: 1=A, 2=B,..., 5=E

#define parameters
param riskCoeff {i in ProjectIDs}; #risk per dollar invested for each project
param initInv; #initial investment amount in dollars

#Set problem variables
var x {i in ProjectIDs} >= 0; #dollars invested in project i
var xb {i in Years} >= 0; #dollars invested in bank for year i
var investment_return = x[2] + 1.4*x[5] + 1.75*x[4] + 1.06*xb[3] - initInv;

#the following are specified by Fig. 1 in the homework PDF

#set objective function
minimize totalRisk: sum{i in ProjectIDs} x[i] * riskCoeff[i];

#specify constraints
s.t. c1: x[1] + x[3] + x[4] + xb[1] = initInv;
s.t. c2: .3*x[1] + 1.1*x[3] + 1.06 * xb[1] = x[2] + xb[2];
s.t. c3: x[1] + .3*x[2] + 1.06*xb[2] = x[5] + xb[3];

#add project investment limits (see HW2)
s.t. limA: x[1] <= 500000;
s.t. limB: x[2] <= 500000;
s.t. limE: x[5] <= 750000;
```

```
param initInv = 1000000;

param : riskCoeff :=
1       .1
2       .12
3       .05
4       .2
5       .05;
```

Running this model results in:

```
Solving the model we can see that the following investment return can be made:
investment_return = 191016

The total risk accrued is:
totalRisk = 0

This is done using the following investments:
x [*] :=
1  0
2  0
3  0
4  0
5  0
;

xb [*] :=
1    1e+06
2  1060000
3  1123600
;

ampl:
```

These results indicate that minimizing risk results in a solution with no risk: only bank investments. This solution corresponds to a return of $191,016 (after subtracting the initial $1,000,000 investment). As expected, this solution results in a much lower return than the solution from part (i).

(ii) Solve the problem using a scalarized objective function to combine the objectives where λ1 ≥ 0 and λ2 ≥ 0 are the weights associated with returns and risks, respectively, and λ1 + λ2 = 1. Use the values, λ1 = 0, 0.25, 0.5, 0.75, 1.

In order to scalarize our objective function, we define two variables in our model: total_risk and investment_return. These variables hold the values of the objective functions from the previous two parts. The final scalarized objective function is then defined as λ(investment_return) – (1 – λ)(total_risk), where λ is one of the values from the question statement. The AMPL implementation is as follows:

```
#create sets to define variables
set Years = {1,2,3}; #years for bank investments
set ProjectIDs = {1,2,3,4,5}; #ID corresponding to each project: 1=A, 2=B,..., 5=E

#define parameters
param riskCoeff {i in ProjectIDs}; #risk per dollar invested for each project
param initInv; #initial investment amount in dollars

#Set problem variables
var x {i in ProjectIDs} >= 0; #dollars invested in project i
var xb {i in Years} >= 0; #dollars invested in bank for year i

#create variables for each of our objective functions
var investment_return = x[2] + 1.4*x[5] + 1.75*x[4] + 1.06*xb[3] - initInv;
var total_risk = sum{i in ProjectIDs} x[i] * riskCoeff[i];

#create a lambda to scalarize our objective function
param lambda;

#scalarize our objective function
maximize scalarized_objective: lambda * investment_return - (1 - lambda)*total_risk;

#specify constraints
s.t. c1: x[1] + x[3] + x[4] + xb[1] = initInv;
s.t. c2: .3*x[1] + 1.1*x[3] + 1.06 * xb[1] = x[2] + xb[2];
s.t. c3: x[1] + .3*x[2] + 1.06*xb[2] = x[5] + xb[3];

#add project investment limits (see HW2)
s.t. limA: x[1] <= 500000;
s.t. limB: x[2] <= 500000;
s.t. limE: x[5] <= 750000;

#read in data from data file
data HW4/data_Q1a_ii.txt;

#solve the model for each lambda
for {l in {0., .25, .5, .75, 1.}}{
        #create a scalarized objective function; note that risk is negative since it competes with investment return
        let lambda := l;
        solve;

        #display the resulting investment return and investments
        printf "Solving the model we can see that the following profit can be made: \n";
        display investment_return;

        printf "The total risk accrued is: \n";
        display total_risk;

        printf "This is done using a lambda of: \n";
        display lambda;
}
```

Note that the data file used for this problem is identical to the previous problem.

Running this model results in the following output:

```
Solving the model we can see that the following profit can be made:
investment_return = 191016

The total risk accrued is:
total_risk = 0

This is done using a lambda of:
lambda = 0

CPLEX 20.1.0.0: optimal solution; objective 83379
1 simplex iterations (0 in phase I)
Solving the model we can see that the following profit can be made:
investment_return = 446016

The total risk accrued is:
total_risk = 37500

This is done using a lambda of:
lambda = 0.25

CPLEX 20.1.0.0: optimal solution; objective 307325
3 simplex iterations (0 in phase I)
Solving the model we can see that the following profit can be made:
investment_return = 797600

The total risk accrued is:
total_risk = 182950

This is done using a lambda of:
lambda = 0.5

CPLEX 20.1.0.0: optimal solution; objective 552462.5
0 simplex iterations (0 in phase I)
Solving the model we can see that the following profit can be made:
investment_return = 797600

The total risk accrued is:
total_risk = 182950

This is done using a lambda of:
lambda = 0.75
```

```
CPLEX 20.1.0.0: optimal solution; objective 797600
0 simplex iterations (0 in phase I)
Solving the model we can see that the following profit can be made:
investment_return = 797600

The total risk accrued is:
total_risk = 182950

This is done using a lambda of:
lambda = 1
```

Interpreting these results, we can observe that as lambda increases, the risk and investment both increase as well. Note that the solutions for $\lambda = 0.5, 0.75, 1$ are all identical. This implies that a more rigorous investigation should be performed on the interval between 0 and .5. We will instead use the epsilon constraint method for a higher resolution estimation of the Pareto Frontier.

(ii) Use the Ɛ-constraint method to solve the investment problem for 20 values of Ɛ. The Ɛ values that you choose should represent the entire spectrum of risk and return. That is, one solution in the set should correspond to the maximum returns achieved in (i) and one solution should correspond to the minimum risk achieved in (ii).

Recall from parts (i) and (ii) that values for risk in feasible solutions range from 0 to 182950. We then can then modify our model to maximize investment return subject to a total risk constraint which we define to be less than epsilon (Ɛ). This epsilon is somewhere in the range between 1 and 182950. We solve the model with 20 different epsilons in AMPL using the following model code:

```
#create sets to define variables
set Years = {1,2,3}; #years for bank investments
set ProjectIDs = {1,2,3,4,5}; #ID corresponding to each project: 1=A, 2=B,..., 5=E

#initialize model parameters
param initInv; #initial amount to be invested in dollars
param riskCoeff {p in ProjectIDs}; #risk per dollar for each project (for part iv)

#Set problem variables
var x {i in ProjectIDs} >= 0; #dollars invested in project i
var xb {i in Years} >= 0; #dollars invested in bank for year i
var risk = sum{i in ProjectIDs} x[i] * riskCoeff[i]; #total risk from investing in projects

#the following are specified by Fig. 1 in the homework PDF

#set objective function
maximize investment_return: x[2] + 1.4*x[5] + 1.75*x[4] + 1.06*xb[3];

#specify constraints
s.t. c1: x[1] + x[3] + x[4] + xb[1] = initInv;
s.t. c2: .3*x[1] + 1.1*x[3] + 1.06 * xb[1] = x[2] + xb[2];
s.t. c3: x[1] + .3*x[2] + 1.06*xb[2] = x[5] + xb[3];

#add project investment limits (see HW2)
s.t. limA: x[1] <= 500000;
s.t. limB: x[2] <= 500000;
s.t. limE: x[5] <= 750000;

#epsilon constraint
param epsilon;
s.t. epsilonConstraint: risk <= epsilon; #we will vary epsilon from 0 (minimum risk) to 182950 (maximum risk)

#load data
data HW4/data_Q1a_iv.txt

#solve the model for each epsilon increment
for {i in 0..19} {
        let epsilon := i * (182950 / 19); #we want to reach our maximum risk after 20 steps
        solve;

        #display the resulting investment return and investments
        printf "\nWith epsilon %f, we see a total return of: %f \n", epsilon, investment_return;
}
```

Again, the data file is identical to the previous question's. The epsilon values are selected uniformly within the feasible interval. Running this model results in the following output:

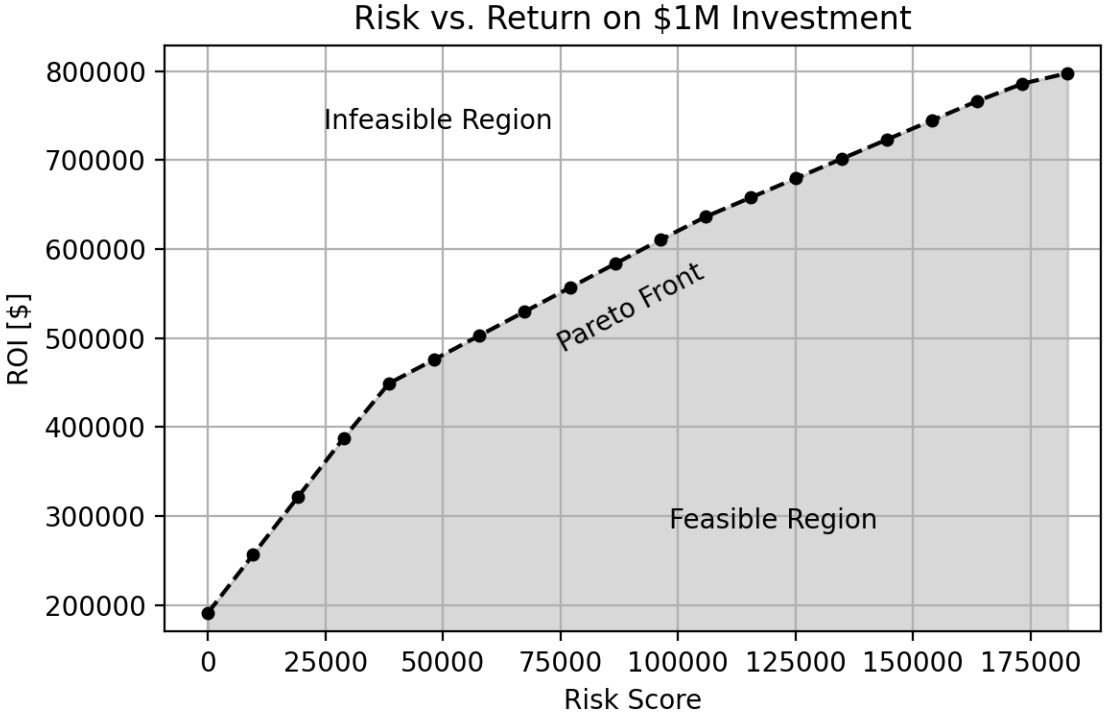
```
Solution determined by presolve;
objective investment_return = 1191016.

With epsilon 0.000000, we see a total return of: 1191016.000000
CPLEX 20.1.0.0: optimal solution; objective 1256492.842
3 dual simplex iterations (1 in phase I)

With epsilon 9628.947368, we see a total return of: 1256492.842105
CPLEX 20.1.0.0: optimal solution; objective 1321969.684
0 simplex iterations (0 in phase I)

With epsilon 19257.894737, we see a total return of: 1321969.684211
CPLEX 20.1.0.0: optimal solution; objective 1387446.526
0 simplex iterations (0 in phase I)

With epsilon 28886.842105, we see a total return of: 1387446.526316
CPLEX 20.1.0.0: optimal solution; objective 1448855.05
1 dual simplex iterations (0 in phase I)

With epsilon 38515.789474, we see a total return of: 1448855.050316
CPLEX 20.1.0.0: optimal solution; objective 1475767.188
0 simplex iterations (0 in phase I)

With epsilon 48144.736842, we see a total return of: 1475767.187895
CPLEX 20.1.0.0: optimal solution; objective 1502679.325
0 simplex iterations (0 in phase I)

With epsilon 57773.684211, we see a total return of: 1502679.325474
CPLEX 20.1.0.0: optimal solution; objective 1529591.463
0 simplex iterations (0 in phase I)

With epsilon 67402.631579, we see a total return of: 1529591.463053
CPLEX 20.1.0.0: optimal solution; objective 1556503.601
0 simplex iterations (0 in phase I)

With epsilon 77031.578947, we see a total return of: 1556503.600632
CPLEX 20.1.0.0: optimal solution; objective 1583415.738
0 simplex iterations (0 in phase I)

With epsilon 86660.526316, we see a total return of: 1583415.738211
CPLEX 20.1.0.0: optimal solution; objective 1610327.876
0 simplex iterations (0 in phase I)

With epsilon 96289.473684, we see a total return of: 1610327.875789
CPLEX 20.1.0.0: optimal solution; objective 1636193.777
1 dual simplex iterations (0 in phase I)

With epsilon 105918.421053, we see a total return of: 1636193.777057
CPLEX 20.1.0.0: optimal solution; objective 1657853.179
0 simplex iterations (0 in phase I)

With epsilon 115547.368421, we see a total return of: 1657853.178636
CPLEX 20.1.0.0: optimal solution; objective 1679512.58
0 simplex iterations (0 in phase I)

With epsilon 125176.315789, we see a total return of: 1679512.580214
CPLEX 20.1.0.0: optimal solution; objective 1701171.982
0 simplex iterations (0 in phase I)

With epsilon 134805.263158, we see a total return of: 1701171.981793
CPLEX 20.1.0.0: optimal solution; objective 1722831.383
```

```
0 simplex iterations (0 in phase I)

With epsilon 144434.210526, we see a total return of: 1722831.383371
CPLEX 20.1.0.0: optimal solution; objective 1744490.785
0 simplex iterations (0 in phase I)

With epsilon 154063.157895, we see a total return of: 1744490.784950
CPLEX 20.1.0.0: optimal solution; objective 1766150.187
0 simplex iterations (0 in phase I)

With epsilon 163692.105263, we see a total return of: 1766150.186528
CPLEX 20.1.0.0: optimal solution; objective 1785752.284
1 dual simplex iterations (0 in phase I)

With epsilon 173321.052632, we see a total return of: 1785752.283922
CPLEX 20.1.0.0: optimal solution; objective 1797600
0 simplex iterations (0 in phase I)

With epsilon 182950.000000, we see a total return of: 1797600.000000
```

With this method, we can see a smoother estimation of the tradeoff between risk and return as opposed to the scalarized objective function method. Note that in the output, epsilon corresponds to the maximum allowed risk score in the solution. These pairs of epsilons and returns were parsed for the next question.

(b) Using the outcomes from either the scalarized or Ɛ-constraint method, graph the Pareto optimal results on objective space. Clearly label the axes, denote the feasible region and infeasible regions, and provide a brief, but clear, interpretation of the results to help the committee understand the graph, the tradeoffs involved, and how to make decisions regarding investments.

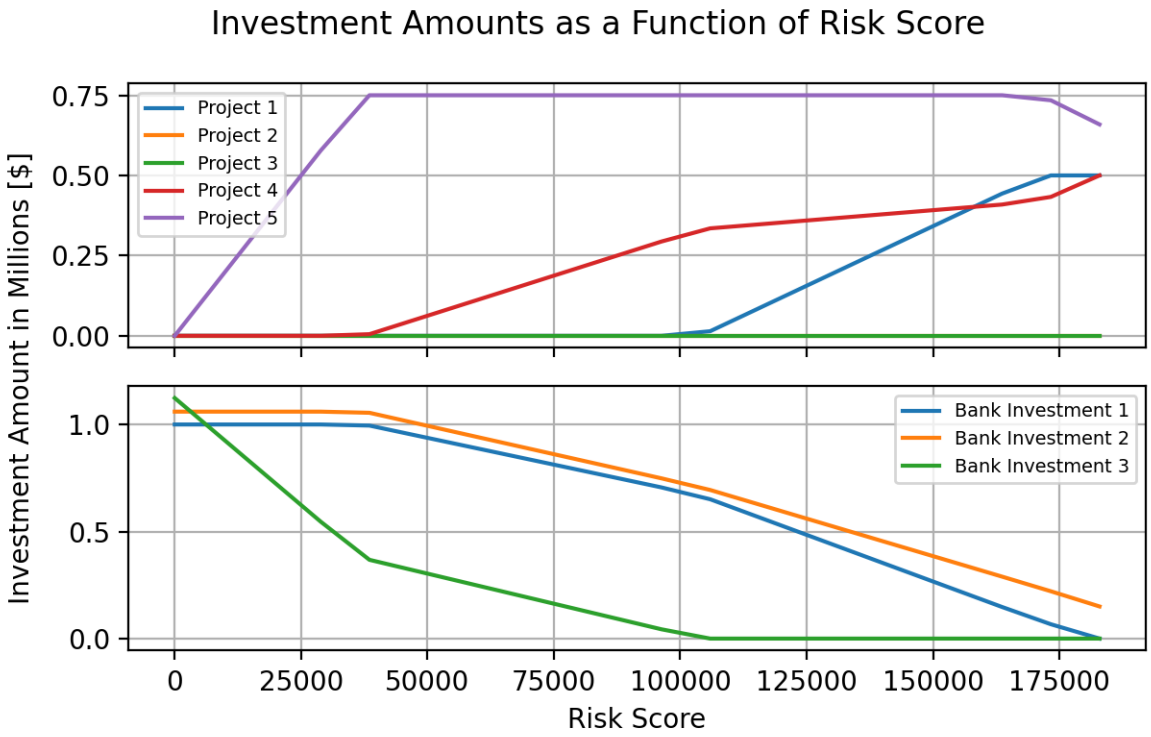
Using the values from part (iv), we wrote a simple python script (zipped) to generate the following visualization:



This graph describes the Pareto Front, or the optimal solutions that form as the result of a tradeoff between weighing the importance of risk vs. return. The dots mark values obtained during part (iv), and the dotted line interpolates linearly between them. Above this boundary, points are infeasible (i.e. there is no investment portfolio that can generate more return for a given risk score). Below this boundary, shaded in grey, is the region of feasible but suboptimal solutions. Between risk scores of 0 and ~37,500, the revenue increases most quickly per unit of risk increase. After this, the revenue increases more slowly per increase in risk after this point. If there is a minimum revenue requirement, then accompanying risk can be found along this Pareto Front.

(c) **Using the Ɛ -constraint results, analyze how the investment portfolio changes across the spectrum of risk-tolerance solutions. Include a clear interpretation to help the decision-makers understand the set of Pareto optimal solutions (visualizations will help!)**

We modified the data display from part (iv) to show the investments for each solution along the Pareto Front. Using these values, we wrote another simple python script (zipped) to generate the following visualization:



We can see that for a risk score of 0, no projects are invested in, with all investment happening in the bank. Project 5 is then shown to be the most profitable investment up until the upper range of risk scores. Bank investment in the 3rd annual period is the initial tradeoff for investment in project 5. Project 3 is the next most appealing option, becoming viable at a risk score of ~37,500 and becoming increasingly important with higher risk tolerance. Investment in this project is traded with investment in all annual bank investments. Finally, Project 1 is shown to become viable at a risk score of ~100,000. Projects 2 and 3 are never shown to be viable solutions.s

Note: Projects 1,2,…,5 are equivalent to Projects A,B,…,E.