# Gerald Jones

# Home Work 3: Capital Budgeting

# ISE522 Spg 22

## Problem Description:

> - Suppose you are an investor, and you are considering investing in **three projects**.
> - It is possible to invest in a fractional amount of a project or the entire amount. For example, if we invest in 0.5 on project 3, then we have cash outflows of -$1 million at time 0 and 0.5.) If you fully invest in a project, the realized cash flows, (in millions of dollars) will be as shown in the following Table.
> - Today we **have $2 million in cash**.
> - At each time period (0, 0.5, 1, 1.5, 2, and 2.5 years from today) we **may**, if desired, **borrow up to $2 million at 3.5\% (per 6 months) interest and must be paid back in the next period. Leftover cash earns 3\% (per 6 months) interest**. For example, if after borrowing and investing at time 0 we have $1 million we would receive $30,000 in interest at time 0.5 years.
> - Your **goal is to maximize cash on hand after accounting for time 3 cash flows**. Formulate an LP to accomplish this goal. Solve using a solver of your choice.

# Notes:

## Parameters:

- Initial on hand cash
- set of investments
- Number of time steps to forcast
- Cash flows expected for each project in given time step
- Loan interest = 3.5% == .35*loan_amount_last_time_step
- Maximum allowable loan amount
- Cash on hand interest == 3% = .3*cash_on_hand_last_time_step
- three parts to a given time periods cash flow/cash on hand amount
    - cash on hand from previous time period
        - i.e. current cash on hand from last time periods time flow
    - cash flow from investments
    - cash flow from loans borrowed
    - cash flow from loans repaided ### Variables and constraints:
- Decision:
    - what proportion of given three choices to invest in
    - when to borrow
    - how much to borrow
- Decision constraints:
    - maximum investment amount (assumed $\leq$ 1 i.e. 100%)
    - maximum allowable borrowing amount
- Lending logic:
    - may borrow >= 2M at 3.5% interest every size months(.5 years or every row/step)
        - must be paid back at next time period
        - pay back amount*interest + amount in next time step
    - we can not owe money at the end of the period i.e. cash flows can not be negative
        - if you will have a negative cash flow at the end of the period borrow enough so that it is not negative up to maximum allowable amount
- Cash on hand logic:
    - cash on hand at time t(current) is based on cash at time t-1(last time step)
        - cash on hand at time t-1 gains interest of the set percentage rate leading to an additional c(*interest_rate) at time t
        - thus cash currently held gains the interest from cash on hand in the previous step and is the sum of the amount of cash on hand at the end of the last time step plus the interest for this amount

# objective:

> Maximize cash on hand to get maximimum cash returns arter the T time steps

IE 60? solving integer programs

## Module imports and data loading

In [1]:
```python
from _GUROBI_TOOLS_.GUROBI_MODEL_BUILDING_TOOLS import *
from _NOTE_BOOK_UTILS import *

data_df = pd.read_excel("HW3_Data.xlsx")
```

## Data Display

In [2]:
```python
print("Data for Optimization Task:")
display(data_df)
print("columns:")
for v in data_df.columns:
    print(v)


# use most negative total investment cost as lower bound for total investment cashflows
Investment_lower_bound = data_df.sum(axis=1).min()
```

```
Data for Optimization Task:
```

| | Years | project 1 | project 2 | project 3 |
|---|---|---|---|---|
| **0** | 0.0 | -3.0 | -2.0 | -2.0 |
| **1** | 0.5 | -1.0 | -0.5 | -2.0 |
| **2** | 1.0 | 1.8 | 1.5 | -1.8 |
| **3** | 1.5 | 1.4 | 1.5 | 1.0 |
| **4** | 2.0 | 1.8 | 1.5 | 1.0 |
| **5** | 2.5 | 1.8 | 0.2 | 1.0 |
| **6** | 3.0 | 5.5 | -1.0 | 6.0 |

```
columns:
Years
project 1
project 2
project 3
```

## Model Formulation

## Parameters:

| | |
|---|---|
| **P** | set of projects that can be invested in where p $\in$ **P** |
| **T** | set of time steps for set time invterval = {0, .5, 1, 1.5, .... , 3}, such that t $\in$ **T** |
| $G$ | interest rate used to determine gains for cash on hand at some time t |
| $L$ | interest rate used to determine losses for money borrowed at some time t |
| $C_0$ | initial cash on hand |
| **R**$_{p,t}$ | cash return for investing in project p at time t |
| $\Lambda$ | maximum allowable loan amount |

### Variables:

Investment decisions and returns/costs

Decision cash flows

| | |
|---|---|
| **X**$_p$ | proportion of investment for project p |
| **A**$_t$ | amount borrowed at time t |

Investment/Interest rate cash flows

| | |
|---|---|
| **O**$_t$ | money owed at time t due to borrowed money at time t-1 |
| **I**$_t$ | profits from cash returns on investments at time t |
| **N**$_t$ | 1 if **I**$_t$ < 0, 0 otherwise |

Cash left over returns

| | |
|---|---|
| **C**$_t$ | cash on hand at the end of time period t |
| **C**$_T$ | cash on hand after T cash flow periods |

## Equations and constraints:

### Proportion of investments constraints

$$0 \leq \quad X_p \quad \leq 1, \forall p$$

$$I_t = \sum_{p=1}^{3}(X_p \cdot R_{p,t}), \forall t$$

Negative investment costs can not exceed the maximum loan amount

$$abs(I_t) \geq \Lambda$$

$$N_t == 0, \implies \quad I_t + A_t + C_{t-1} + C_{t-1} \cdot G - O_t \geq 0$$

### Money borrowed at time t constraints

$$A_t \quad \leq \Lambda$$

### Money owed at time t for loans in t-1:

$$O_t = 0, t = 0$$

$$O_t \quad = A_{t-1} + (A_{t-1} \cdot L), \forall t > 0$$

### Cash on hand/flow at t and after T time steps:

$$C_t > 0, \forall \text{ t}$$

$$C_t = K + A_t + I_t, t = 0$$

$$C_t = I_t + A_t - O_t + C_{t-1} + C_{t-1} \cdot G, \forall t > 0$$

$$C_T = \sum_{t=0}^{T}(C_t)$$

### When to borrow constraint

Have to borrow enough so that the cash flow at time t will not be negative

$$A_t \geq -I_t - (C_{t-1} \cdot L) - C_{t-1} + O_t, \forall t$$

## Objective:

> ### Goal: maximize cash on hand after forcast period (3 years at 6 month intervals):

$$\max((C_T))$$

## Constraint methods

In [3]:
```python
def generate_money_owed_constraints(model, Os, As, l):
    for t in range(len(Os)):
        if t == 0:
            model.addConstr(Os[t] == 0)
        else:
            model.addConstr(Os[t] == As[t-1] + As[t-1]*l)
    return



def generate_profit_by_project_variable_constraint(model, Xs, Ps, df, rate_col_format, col_base=1):
    # for each time step t
    for t in range(len(Ps)):
        # for each project X[p]
        expression = 0
        for p in range(len(Xs)):
            project_number = p+col_base
            print("t{}, p{}, {}".format(t,
                                         rate_col_format.format(project_number),
                                         df.loc[t, rate_col_format.format(project_number)]))
            # for each time step t do : X[p] * R[p, t]
            if p == 0:
                expression = (Xs[p] * df.loc[t, rate_col_format.format(project_number)])
            else:
                expression += (Xs[p] * df.loc[t, rate_col_format.format(project_number)])

        # after all projects have been included for this time step
        # add the constraint for the profit at this time step
        model.addConstr(Ps[t] == expression)
    return
```

## Solve

In [4]:
```python
try:
    ########################################################################################
    ################################## Parameters set up ###################################
    ########################################################################################
    # get the number of time steps
    T = len(data_df)

    # interest on mony left in hand
    G = .03

    # interest rate on borrowed money
    L = .035


    # Initial cash on hand
    K = 2

    # lambda: maximum loan amount
    Lmbda = 2


    print("Parameters:\n\t\t T: {}, G: {}, L:{}, Lambda: {}, Initial Capital: {}".format(T, G, L, Lmbda, K))
    ########################################################################################
    ################################## Variables set up ####################################
    ########################################################################################


    # Create a new model
    m = gp.Model("CapitalBudgeting")


    # #########################################################################################
    # #################################### Investment Based variables #########################
    # #########################################################################################
    # add Investment portions for project variables
    Xp = m.addVars(3, vtype=GRB.CONTINUOUS, name="X", lb=0, ub=1)

    # #########################################################################################
    # #################### Cash flow/on hand at end of time t variables #######################
    # #########################################################################################
    # money on hand variables at time t
    Ct = m.addVars(T, vtype=GRB.CONTINUOUS, name="C")


    # add rate of cash on hand interest variables for project p in time t
    It = m.addVars(T, vtype=GRB.CONTINUOUS, name="I", lb=Investment_lower_bound)



    # #########################################################################################
    # #################################### Loan Based variables ###############################
    # #########################################################################################

    # add amount loan variables for some proportion of the loan for a project
    At = m.addVars(T, vtype=GRB.CONTINUOUS, name="A", lb=0, ub=Lmbda)

    # money owed from loans
    Ot = m.addVars(T, vtype=GRB.CONTINUOUS, name="O")



    ########################################################################################
    ################################## Objective set up ####################################
    ########################################################################################
    #                      Need to maximize the cash on hand at the end of T cash flows
    #                      represented by the last Ct variable
    m.setObjective(Ct[len(Ct)-1], GRB.MAXIMIZE)


    ########################################################################################
    ################################## Constraint set up ####################################
    ########################################################################################
     ##                                    Profits from investments at time t:
    generate_profit_by_project_variable_constraint(m, Xp,
                                                   It,
                                                   data_df,
                                                   rate_col_format="project {}",
                                                   col_base=1)

    ##           Money owed at time t for loans in t-1:
    #            add constraint for loans not occuring if the previous half year had one
    ##           Money borrowed at time t constraints
    generate_money_owed_constraints(m, Ot, At, L)


    ##                                    Cash on hand at t:
    m.addConstr(Ct[0] == K + At[0] + It[0])
    m.addConstrs(Ct[t] == Ct[t-1] + Ct[t-1]*G + At[t] + It[t] - Ot[t] for t in range(1, len(Ct)))


    #                                    add the need to borrow constraint
    m.addConstr(At[0] >= Ot[0] - It[0] - Ct[0] - Ct[0]*G)
    m.addConstrs(At[t] >= Ot[t] - It[t] - Ct[t-1] - Ct[t-1]*G for t in range(1, len(At)))



    ########################################################################################
    ################################## SOLVE:OPTIMIZE ####################################
    ########################################################################################

    m.optimize()
    displayDecisionVars(m, end_sentinel="6")

    print("\n-------------Does it make sense?----------------------")
    print('Obj-Profit after {} time periods: {:.2f}M or ${:,.2f}'.format(T, m.ObjVal, m.ObjVal*1000000))
    print("a {:.0f}% return on the ${}M investment.".format(m.ObjVal*100/K, K))

# catch some math errors
except gp.GurobiError as e:
    print('Error code ' + str(e.errno) + ': ' + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

```
Parameters:
                 T: 7, G: 0.03, L:0.035, Lambda: 2, Initial Capital: 2
Restricted license - for non-production use only - expires 2023-10-25
t0, pproject 1, -3.0
t0, pproject 2, -2.0
t0, pproject 3, -2.0
t1, pproject 1, -1.0
t1, pproject 2, -0.5
t1, pproject 3, -2.0
t2, pproject 1, 1.8
t2, pproject 2, 1.5
t2, pproject 3, -1.8
t3, pproject 1, 1.4
t3, pproject 2, 1.5
t3, pproject 3, 1.0
t4, pproject 1, 1.8
t4, pproject 2, 1.5
t4, pproject 3, 1.0
t5, pproject 1, 1.8
t5, pproject 2, 0.2
t5, pproject 3, 1.0
t6, pproject 1, 5.5
t6, pproject 2, -1.0
t6, pproject 3, 6.0
Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (win64)
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads
Optimize a model with 28 rows, 31 columns and 102 nonzeros
Model fingerprint: 0x1134e006
Coefficient statistics:
  Matrix range     [2e-01, 6e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 7e+00]
  RHS range        [2e+00, 2e+00]
Presolve removed 16 rows and 17 columns
Presolve time: 0.01s
Presolved: 12 rows, 14 columns, 72 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.2399468e+01   2.429021e+01   0.000000e+00      0s
       5    1.2363296e+01   0.000000e+00   0.000000e+00      0s

Solved in 5 iterations and 0.01 seconds (0.00 work units)
Optimal objective  1.236329581e+01
X[0] 0.99027
X[1] 0.00000
X[2] 0.00000
C[0] 0.98522
C[1] 0.00000
C[2] 0.00000
C[3] 1.08881
C[4] 2.90397
C[5] 4.77358
C[6] 12.36330

I[0] -2.97082
I[1] -0.99027
I[2] 1.78249
I[3] 1.38638
I[4] 1.78249
I[5] 1.78249
I[6] 5.44651

A[0] 1.95604
A[1] 2.00000
A[2] 0.28751
A[3] 0.00000
A[4] 0.00000
A[5] 0.00000
A[6] 2.00000

O[0] 0.00000
O[1] 2.02450
O[2] 2.07000
O[3] 0.29757
O[4] 0.00000
O[5] 0.00000
O[6] 0.00000

------------------------------------



-------------Does it make sense?----------------------
Obj-Profit after 7 time periods: 12.36M or $12,363,295.81
a 618% return on the $2M investment.
```

# Solution Discussion:

> From the implemented model the **optimal investment is around .9903 of project 1**, with **borrowed loans of 1.95, 2, and .288, at time steps 0, .5, and 1 respectively**. This solution leads to an **final cash on hand amount of around 12M** for an initial investment of 2M.

# Save Notebook:

## Run the cell below to save the current notebook as a pdf in the same directory

In [5]:
```python
to_PDF("_HW3_CapitalBudgeting.ipynb")
```

```
filename: _HW3_CapitalBudgeting.ipynb
```

In [6]:
```python
currentNotebook
```

Out[6]:
```
''
```