Department of Mathematics and Computer Science
University of Southern Denmark, Odense

April 24, 2018
Marco Chiarandini

DM559/DM545 – Linear and Integer Programming

Obligatory Assignment 1.1, Spring 2018

Solution Contains Solutions!

Preface

Submission deadline:

Friday, April 27, 2018 at noon

The accomplishment of the following assignment is required for the admission to the final written exam.

You have to hand in a report in PDF format containing your answers. The main part of these answers is the description of the models you have used. A proper format, language and mathematical notation as we have learned in class is expected.

You must compile your answers using the LaTeX template provided [tex; pdf]. Those compiling their documents in Word (a choice that is anyway discouraged), should try to stay as close as possible to the template provided. It is very important that you use the structure provided with the clear distinction on the parts of your answers. It is not needed to write a long introduction where you repharse the problem description: it is enough to address directly only the parts required. You can write either in Danish or in English.

You must include in the document the source code of the scripts you have implemented. For those using LaTeX an environment for reporting code is made available in the template. The code must be concise and readable! Source code will however not be checked if the description of the models is not clear and understandable. In other terms, the main focus has to be on the description and mathematical model.

The submission is electronic via:

`http://valkyrien.imada.sdu.dk/milpApp/`

Make sure you have written your name in the first page of the report.

The assignment has to be carried out individually and exchange of solutions is not allowed. However, since the goal of this assignment is to gain experience working with linear programming rather than assessing your preparation, if you are in doubt about how to proceed, it is allowed to consult orally with peers.

Portfolio Selection

Every investor, from the individual to the professional fund manager, must decide on an appropriate mix of assets to include in his or her investment portfolio. The portfolio selection problem consists in deciding, given a sum of money, how much to invest amongst a portfolio of financial securities. The approach that is due to Markowitz (1952) [Mar52] seeks to minimize the risk associated with the investment while realizing a target expected return. By varying the target, one can compute an "efficient frontier", which defines the optimal portfolio for a given expected return.

Let j be an asset from a set of N potential investments, and let R_j denote the random variable indicating the return on (one krone) investment of $j, j = 1, \dots, N$ in the next time period of investment.

A *portfolio* is the fraction of investment to put in each asset in a time period. Hence, a portfolio is determined by a collection of nonnegative numbers $x_j, j = 1, \dots, N$, that sum to one. The return in the next time period that one would obtain from the investment in a portfolio is

$$R = \sum_j x_j R_j$$

and the expected return, henceforth *reward*:[1]

$$E[R] = \sum_j x_j E[R_j]$$

If the reward was the only issue, then the decision would be trivial: simply put everything in the investment with the highest expected return. But unfortunately, investments with high expected return typically also carry a high level of risk. That is, even though they are expected to do very well in the long run, they also tend to be erratic in the short term.

There are many ways to define risk. One way is to define the risk associated with an asset as $x_j(R_j - E[R_j])$ and then for the whole portfolio as the *mean of the absolute deviation from the mean* (MAD):

$$MAD = E\big[|R - E[R]|\big] = E\left[\left|\sum_j x_j \left(R_j - E[R_j]\right)\right|\right]$$

Solving this problem requires knowledge about the joint distribution of the R_j's. This distribution is not known theoretically but it can be estimated looking at historical data. For example, Figure 1 shows monthly returns over a two-year period of a Stock Exchange index and three assets. Let r_{jt} denote the historical return on investment j from month t to month $t+1$ as shown in the tables of Figure 1.[2] One way to estimate the mean $E[R_j]$ is simply to take the average of the historical returns:

$$\hat{R}_j = \frac{1}{T}\sum_{t=1}^{T} r_{jt}.$$

Hence, the estimates for reward $E[R]$ and risk MAD are:

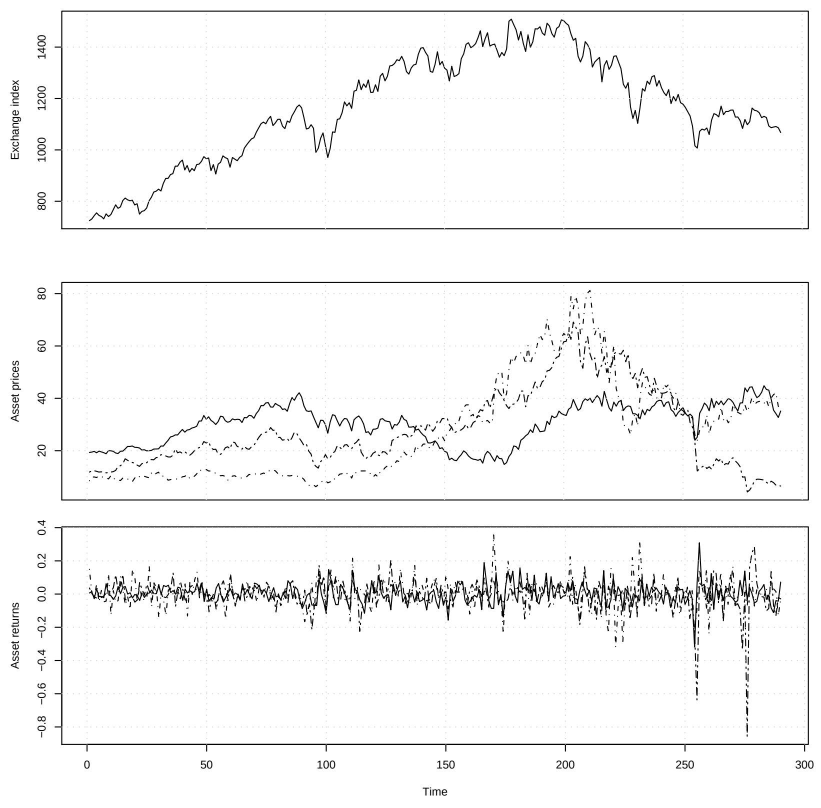
$$\hat{R} = \sum_{j=1}^{N} x_j \hat{R}_j \tag{1}$$

$$\widehat{MAD} = \frac{1}{T}\sum_{t=1}^{T}\left[\left|\sum_{j=1}^{N} x_j \left(r_{jt} - \hat{R}_j\right)\right|\right] \tag{2}$$

[1] We use the symbol E to denote expected value, which means that, if R is a random variable that takes values $R_1, R_2, \dots, R_T$ with equal probability over T time periods, then

$$E[R] = \frac{1}{T}\sum_{t=1}^{T} R_t.$$

[2] Note that we use upper case letter to indicate random variables and lower case letter to indicate their sampled outcome.



c_{jt}	A	B	C
1	19.33	8.52	11.84
2	19.46	9.89	12.28
3	19.75	9.97	12.34
4	19.21	9.75	12.12
5	19.83	10.34	11.84
6	19.54	9.87	11.94
7	19.25	10.09	11.69
8	18.83	9.63	11.56
9	20.04	9.23	11.62
10	19.96	10.43	11.84
11	19.75	9.19	12.00
12	19.12	9.38	12.47
13	18.91	8.92	14.00
14	19.79	8.58	14.25
15	19.83	9.55	15.03

r_{jt}	A	B	C
1	0.01	0.15	0.04
2	0.01	0.01	0.00
3	-0.03	-0.02	-0.02
4	0.03	0.06	-0.02
5	-0.01	-0.05	0.01
6	-0.01	0.02	-0.02
7	-0.02	-0.05	-0.01
8	0.06	-0.04	0.01
9	-0.00	0.12	0.02
10	-0.01	-0.13	0.01
11	-0.03	0.02	0.04
12	-0.01	-0.05	0.12
13	0.05	-0.04	0.02
14	0.00	0.11	0.05
15	0.04	0.02	0.12

Figure 1: The trend of the Stock Exchange index (top), the price (middle) and the returns (bottom) of three investments. The return r_{jt} in a period t is calculated as $\log(c_{j,t-1}) - \log(c_{j,t})$ where c_{jt} is the cost of investment j. (For an explanation of why log returns see: https://quantivity.wordpress.com/2011/02/21/why-log-returns/.) A snapshot of the data for three investments A, B, C in the first 15 months is given in the tables on the right.

The *portfolio selection problem* consists in finding an allocation of assets maximizing the reward while at the same time not incurring in excessive risk. This is a multi-objective situation in which one is confronted with two (or more) competing objectives and there is not a single solution that satisfies them all.

There are several ways to approach multi-objective optimization problems. One approach is scalarization. It consists in defining weights for the objectives and solving only once a single objective problem made by the weighted sum of the objectives. Another approach is ordering lexicographically the objectives and solving a series of single objective problems where objectives of the previous problems become constraints in the next one. These two methods require a priori knowledge, that is, one should be able to discriminate the objectives indicating their relative importance.

A different approach assumes no a priori knowledge. In this case, we search a spectrum of optimal solutions that are indifferent with each other. These solutions compose the set of *Pareto optimal* solutions. More formally, let $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_\ell(\mathbf{x})$, $f_i(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}, i = 1, \ldots, \ell$, be the ℓ objective functions of the problem that map the points of the region of feasible solutions to the objective space. For $\ell = 2$, the situation is illustrated in Figure 2. For a minimization problem, we then say that a feasible solution $\mathbf{x}$ is *dominated* if there exists a feasible solution $\mathbf{x}'$, $\mathbf{x}' \neq \mathbf{x}$ such that $f_i(\mathbf{x}') \leq f_i(\mathbf{x})$ for all $i = 1, \ldots, \ell$ and there is at least one i for which $f_i(\mathbf{x}') < f_i(\mathbf{x})$. A feasible solution $\mathbf{x}$ is *Pareto optimal* if it is not dominated by any other feasible solution. In Figure 2, Pareto optimal solutions are those solutions of the feasible region mapped on the bold frontier in the objective space. The frontier in the objective space determined by Pareto optimal solutions is called *efficient frontier*.

There are different methods to collect the solutions in the Pareto optimal set. One method uses again the scalarization approach but varying the weights throughout and solving a single optimiza-

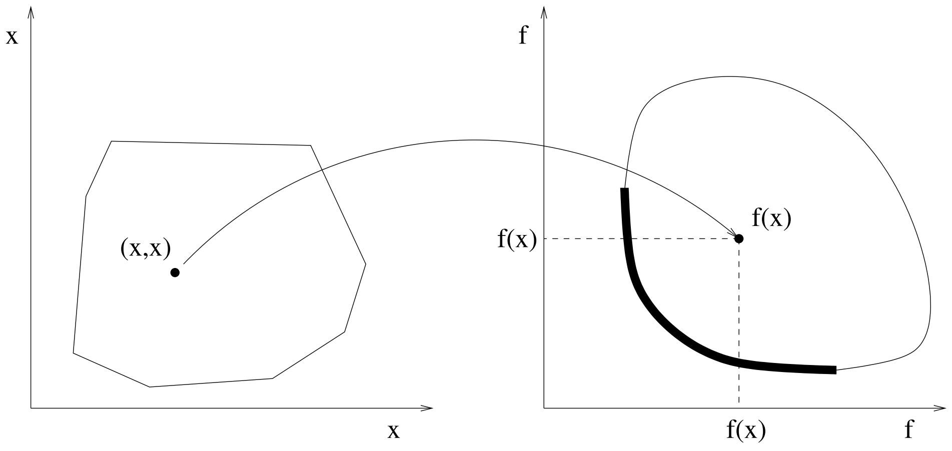


Figure 2: Solution space (left) and objective space (right)

tion problem for each set of weights proposed. Another method, the ϵ-constrained method, optimizes one of the objectives, while using the others as constraints. It proceeds by repeatedly solving single optimization problems varying the ϵ value that is imposed as bound in the objectives used as constraints.

The *portfolio selection problem* is a bi-objective problem in which we are interested in determining the set of Pareto optimal allocations and the efficient frontier with respect to the two criteria: risk and reward.

We will use the ϵ-constraint method for addressing the multi-objective issue, resolving from scratch a new optimization problem every time ϵ is updated. More clever ways exist, avoiding to resolve the problem at each change of ϵ, but for the scope of this assignment this method is well suited. More specifically, in the following, you must treat the reward as the objective to put in the constraint list and the risk as the objective to minimize. The constraint to impose on the reward is that it must be larger than a value B, determined by ϵ in the following way: $B = a + \epsilon(b - a)$ where $a = \max\{0, \min_j \hat{R}_j\}$ and $b = \max_j \hat{R}_j$.

Your Tasks

1. Formulate the portfolio selection problem as a linear programming problem and report the model in mathematical notation. Be precise in your formulation: first, introduce the notation distinguishing parameters and variables, then, write the mathematical model, and, finally, explain each line of the model.

   Further, give the number of variables and constraints that the model has as a function of N and T (note that reporting the numbers indicated by Gurobi is not a correct answer to this question, in that, you have to give the general form as function of N and T).

   Finally, state precisely how many assets at most the optimal solution will indicate as worth buying.

   Solution

   A presentation of the notation would be appropriate here. However, the notation is the same as used in the description, hence I skip it. It may suffice here to state that $x_j \in [0, 1]$, $j = 1..N$ are the variables and all others are given parameters. A non-trivial aspect of the model requested is handling the absolute value in the definition of MAD. We have seen in class that the absolute value is not a linear function. We have also seen how to handle it by introducing auxiliary variables. Here, we introduce a variable $y_t \geq 0$ for each period t.

$$
\begin{aligned}
\min \quad & \frac{1}{T}\sum_{t=1}^{T} y_t && (3)\\
\text{s.t.} \quad & \sum_{j=1}^{N} x_j = 1 && (4)\\
& \sum_{j=1}^{N} x_j(r_{j,t} - E[r_j]) \leq y_t \quad \forall t = 1..T && (5)\\
& \sum_{j=1}^{N} x_j(E[r_j] - r_{j,t}) \leq y_t \quad \forall t = 1..T && (6)\\
& \sum_{j=1}^{N} x_j E[r_j] \geq B && (7)\\
& 0 \leq x_j \leq 1 \quad \forall j = 1..N && (8)\\
& y_t \geq 0 \quad \forall t = 1..T && (9)
\end{aligned}
$$

Constraint (4) ensures the satisfaction of the budget. Constraints (5) and (6) deal with the absolute value, forcing y_t to be equal to the positive value of the left hand side. Note, that y_t is force to be equal and not simply larger because of the objective function that while minimizing the total risk, minimizes all values y_t. Finally, constraint (7) enforces the reward to be larger or equal to the value determined by the threshold ϵ.

As a function of T and N, there are $N+T$ original variables and $2T+2$ constraints (4)-(7) plus N further constraints from the upper bounds of x_j. There are $2T+1+N$ slack variables; hence the total number of variables in the simplex is $N+T+2T+1+N$. From the theory of the simplex and the size of the bases, we known that at most $2T+1+N$ variables (including slack variables) can be strictly larger than zero.

2. Using the template provided portfolio.py, implement the model in Gurobi Python and determine the efficient frontier for the data made available in the file indtrack6.dat for varying values of ϵ. The Python script `portfolio.py` implements the ϵ method providing a function that solves an LP model (the one you have defined) for different values of the argument parameter ϵ. The code produces a plot of the frontier.

   Report and comment relevant information from the run of Gurobi.

   Comment the results described by the plot ensuring that they make sense. Note that, contrary to Figure 2 where we were minimizing both objective functions, here want to minimize the risk while we want to maximize the return; hence the frontier will follow a different pattern than the one of Figure 2. Moreover, state how many assets the model will indicate to buy for varying values of B.

Solution

An implementation in python is given here.

```
def solve(data, epsilon):

    m = Model("portfolio")
    m.setParam(GRB.param.Method, 0)


    B = data.min_r + (epsilon/100)*(data.max_r-data.min_r);

    ### Write here your models
    x={}
    for j in data.assets:
        x[j] = m.addVar(lb=0.0,ub=1.0, obj=0.0, vtype=GRB.CONTINUOUS, name="x_%d" %
             j)
    y={}
    for t in data.times:
        y[t] = m.addVar(lb=0.0,ub=GRB.INFINITY, obj=0.0, vtype=GRB.CONTINUOUS, name
            ="y_%d " % t)

    m.update()

    m.setObjective(quicksum(y[t] for t in data.times), GRB.MINIMIZE)

    m.addConstr(quicksum(x[j] for j in data.assets)==1,"budget")

    for t in data.times:
        m.addConstr(quicksum(x[j]*(data.r[j,t]-data.av_r[j]) for j in data.assets)
            <= y[t], "abs1_%d" % t )
        m.addConstr(quicksum(x[j]*(data.av_r[j]-data.r[j,t]) for j in data.assets)
            <= y[t], "abs2_%d" % t )

    m.addConstr(quicksum( x[j]*data.av_r[j] for j in data.assets) >= B);

    m.optimize()

    count = 0
    for u in x:
        if x[u].x > 0.0001: count +=1
    print "#vars > 0: ",count
    return m.objVal, B
```

The frontier is in Figure 2. We see that, as expected, the larger is the value of the MAD estimate, the higher is the expected reward. The plot represents the trade off between risk and reward. Solutions in the frontier are indifferent. Printing the number of variables strictly larger than 0 for ϵ from 0 to 90 we obtain:

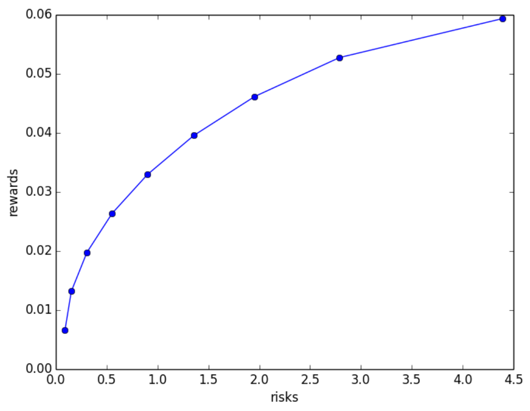
```
#vars > 0:  236
#vars > 0:  227
#vars > 0:  182
#vars > 0:  142
#vars > 0:  104
#vars > 0:  70
#vars > 0:  49
#vars > 0:  26
#vars > 0:  14
```

For $N = 457$ and $T = 290$, this means that also many slack variables will be strictly larger than zero.

3. The following two alternative ways for handling risk instead of minimizing (2) are used in the literature:

   - $\max \min_{t=1,\dots,T} \sum_{j=1}^{n} x_j r_{jt}$, which maximizes the return of the portfolio in the worst period [You98].
   - $\min \hat{\sigma}^2 = \sum_i \sum_j x_i x_j \hat{\sigma}_i \hat{\sigma}_j \rho_{ij}$ where $\hat{\sigma}_j^2 = \frac{1}{T}\sum_{t=1}^{T}\left(r_{jt} - \hat{R}_j\right)^2$ and $\rho_{ij} = 1$ if $i = j$ or 2 otherwise, which leads to the mean-variance portfolio as introduced by Markovitz in the 1950s [Mar52].

   Can these expressions be used in a linear programming problem? For the criteria that can be linearized generate the efficient frontier introducing the new criterion in place of (2) in the



linear programming problem of point 1. Describe the new model, implement it, run the tests and report the new frontier. Comment the results.

Solution

The second expression leads to a quadratic optimization problem and it is not linearazible without loss of information. Gurobi can solve quadratic programming models, see the example at http://www.gurobi.com/documentation/current/examples/portfolio_py.html. However, we do not treat quadratic programming in this course.

The first model is instead linearizable by introducing a single auxiliary variable y as follows.

$$\max \quad y \tag{10}$$

$$\text{s.t.} \sum_{j=1}^{N} x_j = 1 \tag{11}$$

$$\sum_{j=1}^{N} x_j(r_{j,t}) \geq y \quad \forall t = 1..T \tag{12}$$

$$\sum_{j=1}^{N} x_j E[r_j] \geq B \tag{13}$$

$$0 \leq x_j \leq 1 \quad \forall j = 1..N \tag{14}$$

$$y \in \mathbb{R} \tag{15}$$

The python code is:

```
def solve(data, epsilon):

    m = Model("portfolio")
    m.setParam(GRB.param.Method, 0)


    B = data.min_r + (epsilon/100)*(data.max_r-data.min_r);

    ### Write here your models
    x={}
    for j in data.assets:
        x[j] = m.addVar(lb=0.0,ub=1.0, obj=0.0, vtype=GRB.CONTINUOUS, name="x_%d" %
             j)

    y = m.addVar(lb=-GRB.INFINITY,ub=GRB.INFINITY, obj=1.0, vtype=GRB.CONTINUOUS,
        name="y")
```

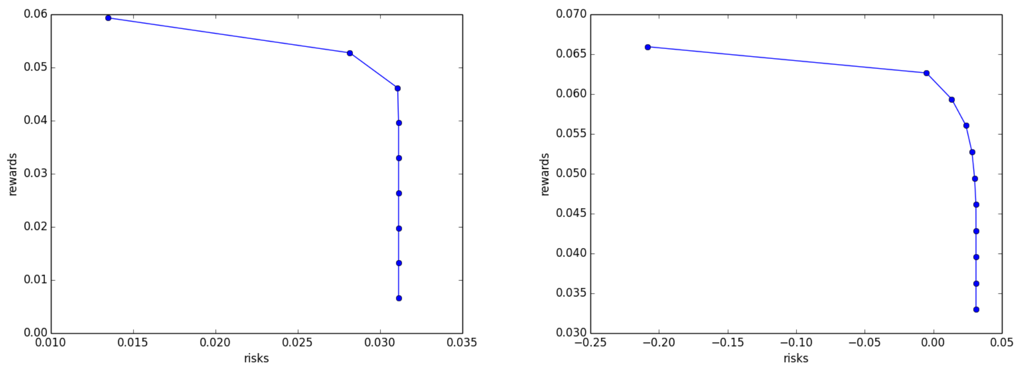


Figure 3:

```
    m.update()

    m.setObjective(y, GRB.MAXIMIZE)

    m.addConstr(quicksum(x[j] for j in data.assets)==1,"budget")

    for t in data.times:
        m.addConstr(quicksum(x[j]*(data.r[j,t]) for j in data.assets) >= y, "
            minimax_%d" % t )

    m.addConstr(quicksum( x[j]*data.av_r[j] for j in data.assets) >= B);

    # m.write("model.lp") # always good to cross check!
    m.optimize()

    return m.objVal, B
```

The frontier in Figure 3 shows again the trade off risk-reward. The two figures use different ϵ sets. For the plot on the left it uses the default values $\{0, 10, 20, \ldots, 90\}$ while the plot on the left zooms on the high reward part with $\epsilon \in \{50, 55, 60, 65, \ldots, 95, 100\}$. Since now the risk is the worst reward through the time periods, as the worst reward decreases (in the picture it becomes even negative, which means we are incurring in a loss) the reward increases. That is, to obtain a high expected return, we have to accept the risk of loosing money in some period.

4. Due to management costs, there is typically a cardinality constraint restricting the number of different assets that can be bought. Model also this aspect in your model from point 1 and solve it for one single point of the frontier, namely for $\epsilon = 20$, when the number of different assets must be smaller than 10. Report the model, the number of variables and constraints expressed in terms of N and T and comment on the computational results.

   Solution We need to introduce binary variables z_j for each $j = 1..N$ that indicates whether we are buying or not the asset and then add two constratins to the model of Task 1:

$$\begin{aligned}
\min \quad & \sum_{t=1}^{T} y_t && (16)\\
\text{s.t.} & \sum_{j=1}^{N} x_j = 1 && (17)\\
& \sum_{j=1}^{N} x_j(r_{j,t} - E[r_j]) \leq y_t \quad \forall t = 1..T && (18)\\
& \sum_{j=1}^{N} x_j(E[r_j] - r_{j,t}) \leq y_t \quad \forall t = 1..T && (19)\\
& \sum_{j=1}^{N} x_j E[r_j] \geq B && (20)\\
& z_j \geq x_j \quad \forall j = 1..N && (21)\\
& \sum_{j=1}^{N} z_j \leq 9 && (22)\\
& 0 \leq x_j \leq 1 \quad \forall j = 1..N && (23)\\
& y_t \geq 0 \quad \forall t = 1..T && (24)\\
& z_j \in \{0,1\} \quad \forall j = 1..N && (25)\\
& && (26)
\end{aligned}$$

We added N variables and $2N+1$ constraints. Since now the problem differently from the previous two tasks is an Interger Linear Programming problem the solution time incereases and we might not be able to find a solution in a reasonable amount of time. I interrupted the execution after 338 nodes had been explored in the branch and bound. At this point the optimality gap was 94.1%, quite far from being closed. The best primal bound was 2.605007439670 and the dual bound 1.527562667533. Since a primal bound was found, a feasible solution was available for use.

5. Another practical issue due to management costs is the presence of a lower bound v to the fraction of assets to put in one single investment. In other terms, the fraction of assets to allocate in one investment can be either zero **or** a value between v and 1. Model this restriction in your model from point 1 and solve it when $v = 0.01$ and $\epsilon = 20$. Report the model, the number of variables and constraints expressed in function of N and T and comment on the computational results.

Solution We introduce again a binary variable z_j for each $j = 1..N$ that indicates whether we are buying or not the asset:

$$\min \quad \sum_{t=1}^{T} y_t \tag{27}$$

$$\text{s.t.} \sum_{j=1}^{N} x_j = 1 \tag{28}$$

$$\sum_{j=1}^{N} x_j(r_{j,t} - E[r_j]) \leq y_t \quad \forall t = 1..T \tag{29}$$

$$\sum_{j=1}^{N} x_j(E[r_j] - r_{j,t}) \leq y_t \quad \forall t = 1..T \tag{30}$$

$$\sum_{j=1}^{N} x_j E[r_j] \geq B \tag{31}$$

$$x_j \leq z_j \quad \forall j = 1..N \tag{32}$$

$$x_i \geq 0.01 z_j \quad \forall j = 1..N \tag{33}$$

$$0 \leq x_j \leq 1 \quad \forall j = 1..N \tag{34}$$

$$y_t \geq 0 \quad \forall t = 1..T \tag{35}$$

$$z_j \in \{0, 1\} \quad \forall j = 1..N \tag{36}$$

$$\tag{37}$$

The python code is:

```
def solve(data, epsilon):

    m = Model("portfolio")
    m.setParam(GRB.param.Method, 0)


    B = data.min_r + (epsilon/100)*(data.max_r-data.min_r);

    ### Write here your models
    x={}
    for j in data.assets:
        x[j] = m.addVar(lb=0.0,ub=1.0, obj=0.0, vtype=GRB.CONTINUOUS, name="x_%d" %
              j)
    y={}
    for t in data.times:
        y[t] = m.addVar(lb=0.0,ub=GRB.INFINITY, obj=0.0, vtype=GRB.CONTINUOUS, name
             ="y_%d " % t)
    z={}
    for j in data.assets:
        z[j] = m.addVar(lb=0.0,ub=1.0, obj=0.0, vtype=GRB.BINARY, name="z_%d" % j)


    m.update()

    m.setObjective(quicksum(y[t] for t in data.times), GRB.MINIMIZE)

    m.addConstr(quicksum(x[j] for j in data.assets)==1,"budget")

    for t in data.times:
        m.addConstr(quicksum(x[j]*(data.r[j,t]-data.av_r[j]) for j in data.assets)
             <= y[t], "abs1_%d" % t )
        m.addConstr(quicksum(x[j]*(data.av_r[j]-data.r[j,t]) for j in data.assets)
             <= y[t], "abs2_%d" % t )
    m.addConstr(quicksum( x[j]*data.av_r[j] for j in data.assets) >= B);

    for j in data.assets:
        m.addConstr(z[j]>=x[j],"z_%d" % j);

    m.addConstr(quicksum( z[j] for j in data.assets) <= 9,"card");

    m.optimize()

    return m.objVal, B
```

We added N variables and $3N + 1$ constraints. Although the problem is like the above one an Integer Programming Problem, Gurobi is able to find the optimal solution already at the root node in less than one seconds. The optimal solution has value $1.601000778778e - 01$.

In the previous editions of this course the glpsolve from GLPK was used. That solver could not find an optimality in less than one hour. An upper bound was found with an optimality gap of 75%:

```
12894: mip =   6.300338790e-01 >=   1.529949735e-01  75.7\% (156; 1)
```

References

[Mar52] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[You98] M.R Young. A minimax portfolio selection rule with linear programming solution. *Management Science*, 44(5):673–683, 1998.