Operations Research II: Algorithms

Gurobi and Python for Nonlinear Programming

Serena Chen

Department of Information Management
National Taiwan University

Road map

- **Introduction**.
- Portfolio optimization.

Introduction

- Besides linear progams and integer programs, there are still many types of problems that Gurobi Optimizer can solve.
- Let's try to solve nonlinear programs with `gurobipy`.
- Recall that a nonlinear model contains interactions between variables.

Road map

- Introduction.
- **Portfolio optimization**.

Portfolio optimization

- Consider the portfolio optimization problem we have introduced in *Operations Research I: Modeling and Applications*.
- We have an opportunity to invest in three different stocks, but our budget is limited.
- Each stock has its own current price, expected future price, and variance of future price.
- Our goal is to decide whether and how much we should invest in each stock in order to minimize the variance (i.e. risk) of total revenue while ensuring the expected revenue is high enough.

Portfolio optimization

- The formulation is

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sigma_i^2 x_i^2 \\ \text{s.t.} \quad & \sum_{i=1}^{n} p_i x_i \leq B \\ & \sum_{i=1}^{n} u_i x_i \geq R \\ & x_i \geq 0 \quad \forall i = 1, ..., n. \end{aligned}$$

The data part

- The data can be set arbitrarily, but now let's follow the settings given in the previous lecture.

```
import pandas as pd

stock_info = pd.read_excel('NLP_dataset.xlsx',
  'Stock information')
stocks = range(len(stock_info['Stock']))
prices = stock_info['Price']
exp_prices = stock_info['Expected price']
variances = stock_info['Variance of the price']

other_info = pd.read_excel('NLP_dataset.xlsx',
  'Budget and min_exp_profit')
budget = other_info['Budget']
min_exp_rev = other_info['Minimum expected profit']
```

- Decouple the data from the model to make your program flexible.

The model part

```
eg3 = Model("eg3")      # build a new model

# add variables as a list
x = []
for i in stocks:
  x.append(eg3.addVar(lb = 0, vtype = GRB.CONTINUOUS,
    name = "x" + str(i+1)))

# setting the objective function
eg3.setObjective(quicksum((variances[i] * x[i] * x[i])
  for i in stocks), GRB.MINIMIZE)

# add constraints and name them
eg3.addConstr(quicksum(prices[i] * x[i]
  for i in stocks) <= budget, "budget_limit")
eg3.addConstr(quicksum(exp_prices[i] * x[i]
  for i in stocks) >= min_exp_rev, "min_revenue")
```

Solving the portfolio optimization problem

- To solve the model and get the solution, type

```
eg3.optimize()
```

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (win64)
Optimize a model with 2 rows, 3 columns and 6 nonzeros
Model fingerprint: 0x84ab935d
Model has 3 quadratic objective terms
Coefficient statistics:
  Matrix range     [2e+01, 6e+01]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e+02, 3e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+05, 1e+05]
Presolve time: 0.01s
Presolved: 2 rows, 3 columns, 6 nonzeros
Presolved model has 3 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 AA' NZ     : 1.000e+00
 Factor NZ  : 3.000e+00
 Factor Ops : 5.000e+00 (less than 1 second per iteration)
 Threads    : 1
```

Solving the portfolio optimization problem

	Objective		Residual			
Iter	Primal	Dual	Primal	Dual	Compl	Time
0	2.03097721e+09	-2.03097721e+09	3.40e+04	2.23e+04	3.15e+08	0s
1	5.66591324e+08	-4.97182349e+08	6.79e+03	7.75e+03	1.24e+08	0s
2	5.92995296e+08	2.34259833e+07	8.91e+03	2.69e+03	8.54e+07	0s
3	3.84752749e+08	2.92382908e+08	2.32e+03	7.07e+02	2.03e+07	0s
4	4.33655425e+08	4.72709996e+08	1.14e+03	4.30e+02	1.90e+07	0s
5	4.95678930e+08	8.14886122e+08	9.23e+02	2.74e+02	1.69e+07	0s
6	1.68275932e+09	1.24036759e+09	0.00e+00	2.74e-04	8.85e+07	0s
7	1.30677756e+09	1.28688757e+09	0.00e+00	4.40e-10	3.98e+06	0s
8	1.28890668e+09	1.28888695e+09	0.00e+00	7.28e-11	3.95e+03	0s
9	1.28888891e+09	1.28888889e+09	0.00e+00	2.91e-11	3.95e+00	0s
10	1.28888889e+09	1.28888889e+09	0.00e+00	2.11e-10	3.95e-03	0s
11	1.28888889e+09	1.28888889e+09	2.55e-12	8.00e-11	3.96e-06	0s
12	1.28888889e+09	1.28888889e+09	3.64e-12	5.45e-11	3.96e-09	0s

```
Barrier solved model in 12 iterations and 0.05 seconds
Optimal objective 1.28888889e+09
```

Solving the portfolio optimization problem

- To see the solution, type

```
for i in stocks:
  print(x[i].varName, '=', x[i].x)

print("z* =", eg3.objVal)  # print objective value

print("Expected profit =", sum(exp_prices[i] * x[i].x
  for i in stocks))
print("Total spending =", sum(prices[i] * x[i].x
  for i in stocks))
```

Solving the portfolio optimization problem

```
x1 = 1333.333333333335
x2 = 833.3333333333313
x3 = 5.2064798070717974e-15
z* = 1288888888.888884
Expected profit = 115000.0
Total spending = 100000.00000000001
```

Some modifications

- Suppose that our goal becomes maximize the total expected revenue given a maximum acceptable risk level (measured by variance).
- We should modify the codes by exchanging the objective function and the second constraint.

```
max_risk = 400000000
eq3.setObjective(quicksum(exp_prices[i] * x[i]
  for i in stocks, GRB.MAXIMIZE)
# ...
eg3.addConstr(quicksum(variance[i] * x[i] * x[i]
  for i in stocks) <= max_risk, "max_risk")
```

- Let the maximum variance of total profit be 400,000,000.
- Run the code we just modify and see how the optimal solution changes.

Some Remarks

- A better way to decouple data from model is to read data from separate data files.
- Gurobi can solve many types of problems.
  - Linear programming (LP).
  - Mixed-integer linear programming (MILP).
  - Quadratic programming (QP).
  - Mixed-integer quadratic programming (MIQP).
  - Quadratically-constrained programming (QCP).
  - Mixed-integer quadratically-constrained programming (MIQCP).
- Gurobi is indeed a powerful solver, but limitations still exist.