MLDS440 Lab 02

Ruiqi Wang

Northwestern University

October 2, 2023

Doctor Assignment Problem

- There are 100 newly graduated doctors who have ranked the type of residency they want to go.
- There are 12 fields of residency, each of which have limited number of available positions.
- Your task is to assign the doctors to proper positions such that the average ranks are minimized.

Problem formulation

- **Index sets**:
  $i \in \mathcal{I}$ represents doctors;
  $j \in \mathcal{J}$ represents fields of residency.

Problem formulation

- **Index sets**:
  $i \in \mathcal{I}$ represents doctors;
  $j \in \mathcal{J}$ represents fields of residency.
- **Data**:
  r_{ij} The rank of field j ranked by doctor i;
  c_j The capacity of field j.

Problem formulation

- **Index sets**:
  $i \in \mathcal{I}$ represents doctors;
  $j \in \mathcal{J}$ represents fields of residency.
- **Data**:
  r_{ij} The rank of field j ranked by doctor i;
  c_j The capacity of field j.
- **Decision variables**: $x_{ij} \in \{0, 1\}$ for whether doctor i is assigned to job j.

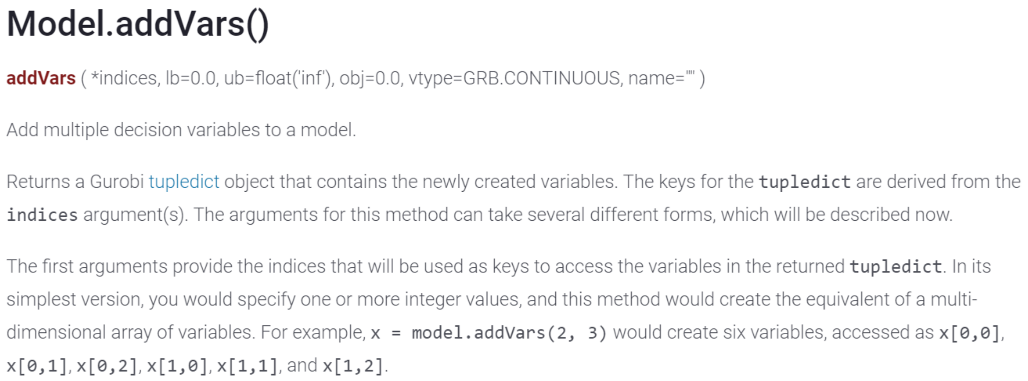
$$\text{Minimize:} \qquad \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} r_{ij} x_{ij}$$

Problem formulation

- **Index sets**:
  $i \in \mathcal{I}$ represents doctors;
  $j \in \mathcal{J}$ represents fields of residency.
- **Data**:
  r_{ij} The rank of field j ranked by doctor i;
  c_j The capacity of field j.
- **Decision variables**: $x_{ij} \in \{0, 1\}$ for whether doctor i is assigned to job j.

$$\begin{aligned}
\text{Minimize:} \quad & \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} r_{ij} x_{ij} \\
\text{Subject to:} \quad & \sum_{i \in \mathcal{I}} x_{ij} \leq c_j \quad \text{for all } j \in \mathcal{J} \\
& \sum_{j \in \mathcal{J}} x_{ij} = 1 \quad \text{for all } i \in \mathcal{I} \\
& x_{ij} \in \{0, 1\} \quad \text{for all } i \in \mathcal{I} \text{ and } j \in \mathcal{J}
\end{aligned}$$

Implementation



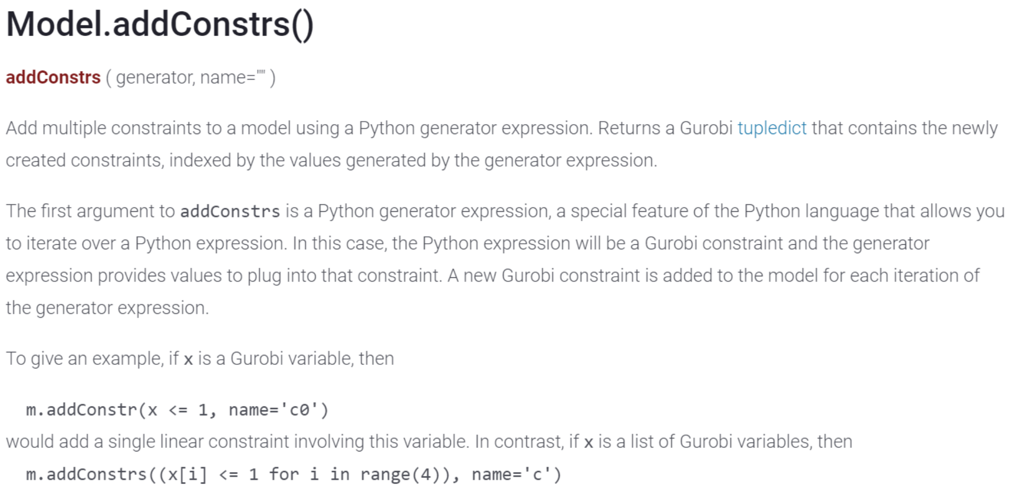
Model.addVars()

addVars (*indices, lb=0.0, ub=float('inf'), obj=0.0, vtype=GRB.CONTINUOUS, name="")

Add multiple decision variables to a model.

Returns a Gurobi tupledict object that contains the newly created variables. The keys for the `tupledict` are derived from the `indices` argument(s). The arguments for this method can take several different forms, which will be described now.

The first arguments provide the indices that will be used as keys to access the variables in the returned `tupledict`. In its simplest version, you would specify one or more integer values, and this method would create the equivalent of a multi-dimensional array of variables. For example, `x = model.addVars(2, 3)` would create six variables, accessed as `x[0,0]`, `x[0,1]`, `x[0,2]`, `x[1,0]`, `x[1,1]`, and `x[1,2]`.



Model.addConstrs()

addConstrs (generator, name="")

Add multiple constraints to a model using a Python generator expression. Returns a Gurobi tupledict that contains the newly created constraints, indexed by the values generated by the generator expression.

The first argument to `addConstrs` is a Python generator expression, a special feature of the Python language that allows you to iterate over a Python expression. In this case, the Python expression will be a Gurobi constraint and the generator expression provides values to plug into that constraint. A new Gurobi constraint is added to the model for each iteration of the generator expression.

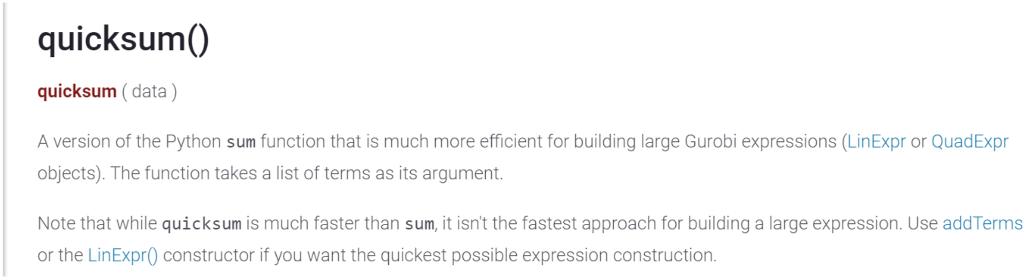
To give an example, if `x` is a Gurobi variable, then

```
  m.addConstr(x <= 1, name='c0')
```

would add a single linear constraint involving this variable. In contrast, if `x` is a list of Gurobi variables, then

```
  m.addConstrs((x[i] <= 1 for i in range(4)), name='c')
```

Quicksum



quicksum()

quicksum (data)

A version of the Python `sum` function that is much more efficient for building large Gurobi expressions (LinExpr or QuadExpr objects). The function takes a list of terms as its argument.

Note that while `quicksum` is much faster than `sum`, it isn't the fastest approach for building a large expression. Use addTerms or the LinExpr() constructor if you want the quickest possible expression construction.

Extension

Consider adding the following constraints to the original problem

Extension

Consider adding the following constraints to the original problem

- The doctors graduated from four different medical schools. The hospital wants to control the minimum and maximum number of doctors from each school in each position.

Extension

Consider adding the following constraints to the original problem

- The doctors graduated from four different medical schools. The hospital wants to control the minimum and maximum number of doctors from each school in each position.
- School 3 wants at least 10 of their students go to job 1 to 5. How to make sure that happens?

Extension

Consider adding the following constraints to the original problem

- The doctors graduated from four different medical schools. The hospital wants to control the minimum and maximum number of doctors from each school in each position.
- School 3 wants at least 10 of their students go to job 1 to 5. How to make sure that happens?
- We definitely don't want anyone to be extremely disappointed. Let us make sure that each doctor is not assigned to their worst d options. How to make that happen? What is the maximal feasible value of d?

Extension

- **New index sets**: $k \in \mathcal{K}$ represents schools
- **New data**:
  $s_{ik} \in \{0, 1\}$ Whether doctor i graduated from school k.
  m_{kj}, M_{kj} The minimum and maximum constraint of total number from school k for job j.
- **New constraints**:
  School-job constraints: $m_{kj} \leq \sum_{i \in \mathcal{I}} s_{ik} x_{ij} \leq M_{kj}$ for all $j \in \mathcal{J}$ and $k \in \mathcal{K}$

Extension

- **New index sets**: $k \in \mathcal{K}$ represents schools
- **New data**:
  $s_{ik} \in \{0, 1\}$ Whether doctor i graduated from school k.
  m_{kj}, M_{kj} The minimum and maximum constraint of total number from school k for job j.
- **New constraints**:
  School-job constraints: $m_{kj} \leq \sum_{i \in \mathcal{I}} s_{ik} x_{ij} \leq M_{kj}$ for all $j \in \mathcal{J}$ and $k \in \mathcal{K}$
  First 5 jobs for school 3: $\sum_{i \in \mathcal{I}} \sum_{j=1}^{5} s_{i3} x_{ij} \geq 10$

Extension

- **New index sets**: $k \in \mathcal{K}$ represents schools
- **New data**:
  $s_{ik} \in \{0, 1\}$ Whether doctor i graduated from school k.
  m_{kj}, M_{kj} The minimum and maximum constraint of total number from school k for job j.
- **New constraints**:
  School-job constraints: $m_{kj} \leq \sum_{i \in \mathcal{I}} s_{ik} x_{ij} \leq M_{kj}$ for all $j \in \mathcal{J}$ and $k \in \mathcal{K}$
  First 5 jobs for school 3: $\sum_{i \in \mathcal{I}} \sum_{j=1}^{5} s_{i3} x_{ij} \geq 10$
  Min-max the rank: $\sum_{j \in \mathcal{J}} r_{ij} x_{ij} \leq 7$ for all $i \in \mathcal{I}$