# Homework Assignment 3 Fall 2024

**IE 523 Financial Computing** Due on Wednesday 10/16 (at 11:59 pm)

In the following three questions, you are asked to mathematically model, implement in `Gurobi`, and then solve three slightly different versions of so-called assignment problems. In all three questions, I am expecting a full mathematical formulation with clear definitions of sets, data, decision variables, and a format that follows:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq b \\ & x \in X. \end{aligned}$$

## Question 1: Simple linear assignments

One of the most common optimization problems that is solved daily is the *assignment problem*. To be exact, given two sets of elements $I$ and $J$, the goal is to match each element in $I$ to at most one element in $J$, and each element in $J$ to at most one element in $I$. When the two sets are of equal cardinalities, then the goal is to match each element in $I$ to exactly one element in $J$, and each element in $J$ to exactly one element in $I$. If $|I| = n$ and $|J| = m$, then without loss of generality you may assume that $I = \{1, 2, \ldots, n\}$, and $J = \{1, 2, \ldots, m\}$.

It is common to assume that there is a cost function (relationship) when assigning $i$ to $j$: let it be $c_{ij}$. You may assume that this cost is always non-negative, for the purposes of this exercise.

**Formulate the assignment problem described as a mathematical program.** That is, you need to describe the data, sets, decision variables, constraints, and objective function in the usual form we saw in class. **Then, write an `Assignment` class in C++ that describes an instance of an assignment problem. The class should have a `read(filename)` functionality that can read the costs of an assignment as in the file linear_assignment.dat (see also Figure 1). The class should also have a `solve()` functionality that identifies the optimal (minimum cost) assignment. This last part should be coded using Gurobi.**

$$\boxed{\begin{array}{llll} n & m & & \\ c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nm} \end{array}}$$

Figure 1: The formatting of the file containing the linear assignment data.

## Question 2: Generalized assignment problems

Let us extend the previous definition to bring along a *third* set: let $K$ be another set of elements. Furthermore, assume that we now have a new cost definition in the form of $c_{ijk}$ which assigns elements $i \in I, j \in J, k \in K$ together. Like earlier, our goal is to find a feasible assignment (i.e., assigning each element in $I$ to at most one element in $J$, each element in $I$ to at most one element in $K$, and so on), that minimizes the total cost of assignment.

Moreover, assume that we have $d$ sets $(I_1, I_2, \ldots, I_d)$ that we aim to assign each element in $I_\ell$ to at least one element in every other set. For convenience, we assume that $|I_1| = |I_2| = \ldots = |I_d| = n$.

**Formulate this new, generalized (multidimensional) assignment problem described as a mathematical program. Then, write an `MultiAssignment` class in C++ that describes an instance of such an assignment problem. The class should have a `read(filename)` functionality that can read the costs of an assignment as in the file multidimensional_assignment.dat (see also Figure 2). The class should also have a `solve()` functionality that identifies the optimal (minimum cost) assignment. This last part should be coded using Gurobi.**

| | | |
|---|---|---|
| 4 | 3 | |
| $c_{1111}$ | $c_{1112}$ | $c_{1113}$ |
| $c_{2111}$ | $c_{2112}$ | $c_{2113}$ |
| $c_{3111}$ | $c_{3112}$ | $c_{3113}$ |
| $c_{1211}$ | $c_{1212}$ | $c_{1213}$ |
| $c_{2211}$ | $c_{2212}$ | $c_{2213}$ |
| $c_{3211}$ | $c_{3212}$ | $c_{3213}$ |
| $c_{1311}$ | $c_{1312}$ | $c_{1313}$ |
| $c_{2311}$ | $c_{2312}$ | $c_{2313}$ |
| $c_{3311}$ | $c_{3312}$ | $c_{3313}$ |
| $c_{1121}$ | $c_{1122}$ | $c_{1123}$ |
| $c_{2121}$ | $c_{2122}$ | $c_{2123}$ |
| $c_{3121}$ | $c_{3122}$ | $c_{3123}$ |
| $c_{1221}$ | $c_{1222}$ | $c_{1223}$ |
| $c_{2221}$ | $c_{2222}$ | $c_{2223}$ |
| $c_{3221}$ | $c_{3222}$ | $c_{3223}$ |
| $c_{1321}$ | $c_{1322}$ | $c_{1323}$ |
| $c_{2321}$ | $c_{2322}$ | $c_{2323}$ |
| $c_{3321}$ | $c_{3322}$ | $c_{3323}$ |
| $c_{1131}$ | $c_{1132}$ | $c_{1133}$ |
| $c_{2131}$ | $c_{2132}$ | $c_{2133}$ |
| $c_{3131}$ | $c_{3132}$ | $c_{3133}$ |
| $c_{1231}$ | $c_{1232}$ | $c_{1233}$ |
| $c_{2231}$ | $c_{2232}$ | $c_{2233}$ |
| $c_{3231}$ | $c_{3232}$ | $c_{3233}$ |
| $c_{1331}$ | $c_{1332}$ | $c_{1333}$ |
| $c_{2331}$ | $c_{2332}$ | $c_{2333}$ |
| $c_{3331}$ | $c_{3332}$ | $c_{3333}$ |

Figure 2: The formatting of the file containing the multidimensional assignment data. Here we saw a $d = 4$, $n = 3$ example. Note that you should always expect the file to contain $n^d$ costs – that is, in the above example there exist $3^4 = 81$ costs.

### Question 3: Quadratic assignment problems

The last problem we discuss is the quadratic version of the assignment problem. Assume that we are given four set $I, J, K, L$ (such that they are all of the same cardinality), and two cost functions $c_{ij}, \forall i \in I, j \in J$ and $d_{k\ell}, \forall k \in K, \ell \in L$. To complicate matters though, the goal is to assign each element in $I$ to exactly one element in $K$ (and vice versa) – not $J$! Similarly, we want to simultaneously assign each element in $J$ to exactly one element in $L$ (and vice versa).

It may help to talk about an application. Let $I$ be a set of people, $J$ a set of jobs, $K$ a set of offices, and $L$ a set of equipment. The cost of assigning a person $i$ to a job $j$ is $c_{ij}$; the distance of an office $k$ to some equipment $\ell$ is $d_{k\ell}$. If we assign person $i$ to office $k$, and job $j$ to use equipment $\ell$, then we need to pay $c_{ij} \cdot d_{k\ell}$ units to have person $i$ do job $j$ starting from office $k$ and using equipment $\ell$. Notice that we do not assign persons to jobs; nor equipments to offices! Instead, we assign people to offices and equipment to jobs. **Please email me if you need more hints**.

**Formulate the assignment problem described as a mathematical program.** No need for C++ code or Gurobi. Simply the mathematical program (data, sets, decision variables, constraints, and objective function) will do.