IEM 5013 Introduction to Optimization
Course Project

Assignment Policy: This assignment can be done by teams of up to **three** students. A student can be part of exactly one submission. In order to include your name in the team's submission you must have made a substantial contribution by actively and continuously participating in the development of your team's assignment submission.

Permitted Resources: Course textbook and any other textbook, even if it is not listed among the references on the course syllabus. All materials provided on the Canvas course website. All materials available at Gurobi.com and at other links provided on the Canvas course website as learning resources for the Python programming language.

Prohibited Activities: Collaboration, using any means, with anyone other than your teammate. Searching the internet for answers. Accessing past assignment solutions with the help of former students or using websites that facilitate this access. Use of any resource outside the resources explicitly noted above as permissible. Anything else I haven't enumerated, but violates the spirit of academic integrity.

The assignment must be completed using only your individual/team effort and no external assistance beyond the permissible resources.

ATCO Scheduling Problem

In this project, we will consider a simplified version of a scheduling problem. Specifically, the scheduling of Air Traffic Controllers (ATCOs) who work for the Federal Aviation Authority to help safely guide aircrafts inside the national air space. Consider a 2-week planning horizon corresponding to one pay period, split into 15-minute time intervals. This gives a planning horizon with T time periods, where $T = 4 \times 24 \times 14 = 1,344$. In each time period $t = 1, 2, \ldots, T$, at least r_t ATCOs are required on duty.

Two types of ATCO schedules are permitted by contractual agreements for an ATCO: (a) 10 $\times$ 8-hour shifts per schedule; (b) 8 $\times$ 10-hour shifts per schedule. We refer to these as the *regular* and *compressed* work schedules, respectively. Based on fatigue rules, there must be a gap of at least 8 hours between consecutive shifts in a schedule. In practice, this rest period can depend on the time of day (e.g., 8 hours after a day shift, 9 hours after an evening shift, 12 hours after a midnight shift), but for now we assume it is a constant 8 hours. Overtime can be handled as a short extension of a shift, or as a standalone extra shift, the ATCO who is assigned that schedule is asked to cover. For simplicity, all ATCOs can be assumed to be paid the same hourly rate, and overtime increases the hourly rate by a factor of 1.50. There are two optimization problems we can consider in this setting, that are described next.

Shift distribution problem. This problem asks us to find the minimum total number of 8-hour shifts and 10-hour shifts needed to cover the staffing requirement r_t in each time period. Note that we do not build a complete work schedule (regular or compressed) for an ATCO in this model and no overtime extension is permitted. We simply seek an "optimal distribution" of shifts—the number shifts of each type, and when each must start and finish. There are heuristic methods that then try to combine these shifts into schedules that is outside our scope.

Shift scheduling problem. This problem asks us to find a minimum total cost collection of regular and compressed schedules that meet the coverage requirement r_t in each time period t of the planning horizon. Note that in this problem we are building a fixed (user-specified) number of complete (regular and compressed) schedules, with overtime as needed, that ATCOs will be assigned to work (in reality they bid on their preferred schedules). This is a complete solution.

Assignment.

1. (75 points) Formulate and implement an integer programming approach using Gurobi and Python to solve the shift distribution problem.
2. (25 points) Formulate and implement an integer programming approach using Gurobi and Python to solve the shift scheduling problem. This is a considerably more challenging problem. Please attempt this only if you have completed the previous assignment to your complete satisfaction.

Please note the following.

- The data files providing the requirements r_t are posted on Canvas.
- Use the Gurobi parameter for terminating the algorithm early based on a **user-specified time limit** to obtain a good solution with an MIP gap in a reasonable amount of time. You can also use the OSU supercomputer Pete if you have prior experience. Gurobi is installed on the cluster. You can simplify the shift scheduling problem in reasonable ways to help with solution.
- Do not modify the instances in anyway; your code must read the instances as is, given the filename as input. We will validate your code using these and other identically formatted instances. Test your implementation on all instances before submission.
- A Python file named `LastName1_LastName2_LastName3.py` after the team members, clearly commented and formatted, should be uploaded to Canvas. Additionally, list the team member names at the start of the code file as comments.
- Check the **termination status** of the algorithm before writing the output. Write your output to an external file—first line/header must contain the termination status (optimal, feasible), best objective value, optimality gap at termination, and elapsed wall-clock time. Time permitting, report the actual schedules you created in a manner that you think is useful for the scheduler to read.