ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF ENGINEERING – DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF ELECTRONICS AND COMPUTERS

OPERATIONAL RESEARCH 2025

KARATIS DIMITRIOS 10775

INTRODUCTION

Libraries

The following tasks of the assignment will be implemented and solved using **Gurobi Optimization for Python**, which is a mathematical optimization software library for the **Python 3** programming language, designed to solve optimization problems.

In [2]:
```
%%capture
!pip install gurobipy
```

In [3]:
```
import gurobipy as gp
from gurobipy import Model, GRB, quicksum
```

Additionally, the following libraries were used as auxiliary tools:

In [4]:
```
%%capture
!pip install pandas
!pip install numpy
```

In [5]:
```
import pandas as pd
import numpy as np
```

QUESTION 1

Course Scheduling Problem Modeling

Sets

- $T = \{1, 2\}$: Set of classes
- $D = \{\text{Mon}, \text{Tue}, \text{Wed}, \text{Thu}, \text{Fri}\}$: Set of days
- $Z = \{1, 2, 3, 4\}$: Set of time slots per day
- M: Set of courses
- P: Set of professors

Parameters

- $r_{m,t}$: Number of two-hour sessions that must be taught for course m in class t during the week
- $a_m \in P$: Professor teaching course m

Decision Variables

- $x_{m,t,d,z} \in \{0, 1\}$:
  - $x_{m,t,d,z} = 1$ if course m of class t is taught on day d in slot z
  - $x_{m,t,d,z} = 0$ otherwise

Objective Function

No cost minimization/maximization is required — the problem is a feasibility problem:

$$\min 0$$

Constraints

1. Correct number of sessions per course/class

$$\sum_{d \in D} \sum_{z \in Z} x_{m,t,d,z} = r_{m,t} \quad \forall m, t$$

---

2. At most one course per slot for each class

$$\sum_{m \in M} x_{m,t,d,z} \leq 1 \quad \forall t \in T, d \in D, z \in Z$$

---

3. Each professor teaches at most one time slot

$$\sum_{m \in M: a_m = p} (x_{m,1,d,z} + x_{m,2,d,z}) \leq 1 \quad \forall p \in P, d \in D, z \in Z$$

---

4. Physical Education only on Thursday afternoon (slot 3)

For each Physical Education course:

$$x_{m,t,d,z} = 0 \quad \forall m = \text{P.E.}, \text{if } d \neq \text{Thu or } z \neq 3$$

---

5. Mr. Lathopraxis does not teach Monday morning (slot 1)

$$x_{\text{Math2},2,\text{Mon},1} = 0$$

---

6. Ms. Insulina does not teach on Wednesday

$$x_{m,t,\text{Wed},z} = 0 \quad \forall z \in Z, \forall t, \text{where } a_m = \text{Insulina}$$

---

7. Each course can occur at most once per day per class

$$\sum_{z \in Z} x_{m,t,d,z} \leq 1 \quad \forall m, t, d$$

---

8. The first time slot on Monday is blocked

$$\sum_{m \in M} x_{m,t,\text{Mon},1} = 0 \quad \forall t \in T$$

Gurobi code

In [6]:

```
# Problem Data
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
slots = [1, 2, 3, 4]  # 8:00-10:00, 10:15-12:15, 14:00-16:00, 16:15-18:15
sections = [1, 2]  # Section 1 and Section 2

teachers = {
    'Gesmanidis': ['English'],
    'Insulina': ['Biology'],
    'Chartoula': ['History-Geography'],
    'Lathopraxis': ['Math2'],
    'Antiparagogos': ['Math1'],
    'Kirkofidou': ['Physics'],
```

```python
    'Platiazon': ['Philosophy'],
    'Bratsakis': ['PE1'],
    'Trekhalitoula': ['PE2']
}

# Lesson data: (Teacher, Section, Hours per week)
lessons = {
    ('English', 1): ('Gesmanidis', 1),
    ('English', 2): ('Gesmanidis', 1),
    ('Biology', 1): ('Insulina', 3),
    ('Biology', 2): ('Insulina', 3),
    ('History-Geography', 1): ('Chartoula', 2),
    ('History-Geography', 2): ('Chartoula', 2),
    ('Math1', 1): ('Antiparagogos', 4),
    ('Math2', 2): ('Lathopraxis', 4),
    ('Physics', 1): ('Kirkofidou', 3),
    ('Physics', 2): ('Kirkofidou', 3),
    ('Philosophy', 1): ('Platiazon', 1),
    ('Philosophy', 2): ('Platiazon', 1),
    ('PE1', 1): ('Bratsakis', 1),
    ('PE2', 2): ('Trekhalitoula', 1)
}

# Create model
model = Model("SchoolTimetable")

# Decision variables: x[lesson, section, day, slot] ∈ {0,1}
x = {}
for (lesson, sec), (teacher, hours) in lessons.items():
    for d in days:
        for z in slots:
            x[lesson, sec, d, z] = model.addVar(vtype=GRB.BINARY, name=f"x_{

model.update()

# Constraint 1: Number of lessons per course/section
for (lesson, sec), (teacher, hours) in lessons.items():
    model.addConstr(quicksum(x[lesson, sec, d, z] for d in days for z in slo

# Constraint 2: At most one lesson per slot for each section
for sec in sections:
    for d in days:
        for z in slots:
            model.addConstr(quicksum(x[lesson, sec, d, z] for (lesson, s), v

# Constraint 3: Each teacher cannot teach two sections at the same time
for teacher in teachers:
    for d in days:
        for z in slots:
            model.addConstr(
                quicksum(x[lesson, sec, d, z] for (lesson, sec), (t, h) in l
            )

# Constraint 4: PE only on Thursday afternoon (slot 3)
for (lesson, sec), (teacher, hours) in lessons.items():
    if lesson in ['PE1', 'PE2']:
```

```python
        for d in days:
            for z in slots:
                if not (d == 'Thu' and z == 3):
                    model.addConstr(x[lesson, sec, d, z] == 0)

# Constraint 5: Lathopraxis does not teach Monday morning (slot 1)
model.addConstr(x['Math2', 2, 'Mon', 1] == 0)

# Constraint 6: Insulina does not teach on Wednesday
for z in slots:
    for (lesson, sec), (teacher, h) in lessons.items():
        if teacher == 'Insulina':
            model.addConstr(x[lesson, sec, 'Wed', z] == 0)

# Constraint 7: Each lesson occurs at most once per day per section
for (lesson, sec), (teacher, h) in lessons.items():
    for d in days:
        model.addConstr(quicksum(x[lesson, sec, d, z] for z in slots) <= 1)

# Constraint 8: First slot on Monday is blocked
for sec in sections:
    for (lesson, s), _ in lessons.items():
        if s == sec:
            model.addConstr(x[lesson, sec, 'Mon', 1] == 0)

# Objective: feasibility, so just optimize
model.setObjective(0, GRB.MINIMIZE)

# Solve
model.optimize()

# Display solution
if model.Status == GRB.OPTIMAL:
    print("\nFinal Timetable:")
    for (lesson, sec, d, z), var in x.items():
        if var.X > 0.5:
            print(f"Section {sec} - {lesson} -> {d} slot {z}")
else:
    print("No feasible solution found.")

# Display timetable as table for each section
if model.Status == GRB.OPTIMAL:
    for sec in sections:
        timetable = pd.DataFrame('', index=days, columns=slots)
        for (lesson, s, d, z), var in x.items():
            if var.X > 0.5 and s == sec:
                timetable.at[d, z] = lesson
        print(f"\n Section {sec}")
        display(timetable)
else:
    print("No feasible solution found.")
```

```
Restricted license - for non-production use only - expires 2026-11-23
Gurobi Optimizer version 12.0.3 build v12.0.3rc0 (linux64 - "Ubuntu 22.04.4
LTS")

CPU model: Intel(R) Xeon(R) CPU @ 2.20GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 1 physical cores, 2 logical processors, using up to 2 threads

Optimize a model with 365 rows, 280 columns and 1181 nonzeros
Model fingerprint: 0xfe6e3487
Variable types: 0 continuous, 280 integer (280 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [0e+00, 0e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+00]
Found heuristic solution: objective 0.0000000

Explored 0 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 1 (of 2 available processors)

Solution count 1: 0

Optimal solution found (tolerance 1.00e-04)
Best objective 0.000000000000e+00, best bound 0.000000000000e+00, gap 0.000
0%

Final Timetable:
Section 1 - English -> Fri slot 3
Section 2 - English -> Thu slot 4
Section 1 - Biology -> Mon slot 4
Section 1 - Biology -> Thu slot 1
Section 1 - Biology -> Fri slot 4
Section 2 - Biology -> Mon slot 3
Section 2 - Biology -> Tue slot 2
Section 2 - Biology -> Fri slot 1
Section 1 - History-Geography -> Tue slot 1
Section 1 - History-Geography -> Thu slot 2
Section 2 - History-Geography -> Mon slot 2
Section 2 - History-Geography -> Fri slot 4
Section 1 - Math1 -> Mon slot 3
Section 1 - Math1 -> Tue slot 2
Section 1 - Math1 -> Thu slot 4
Section 1 - Math1 -> Fri slot 1
Section 2 - Math2 -> Mon slot 4
Section 2 - Math2 -> Wed slot 3
Section 2 - Math2 -> Thu slot 2
Section 2 - Math2 -> Fri slot 2
Section 1 - Physics -> Mon slot 2
Section 1 - Physics -> Tue slot 3
Section 1 - Physics -> Wed slot 1
Section 2 - Physics -> Tue slot 4
Section 2 - Physics -> Wed slot 4
Section 2 - Physics -> Fri slot 3
Section 1 - Philosophy -> Wed slot 2
Section 2 - Philosophy -> Wed slot 1
Section 1 - PE1 -> Thu slot 3
```

```
Section 2 - PE2 -> Thu slot 3
```

```
Section 1
```

	1	2	3	4
Mon		Physics	Math1	Biology
Tue	History-Geography	Math1	Physics	
Wed	Physics	Philosophy		
Thu	Biology	History-Geography	PE1	Math1
Fri	Math1		English	Biology

```
Section 2
```

	1	2	3	4
Mon		History-Geography	Biology	Math2
Tue		Biology		Physics
Wed	Philosophy		Math2	Physics
Thu		Math2	PE2	English
Fri	Biology	Math2	Physics	History-Geography

Solution Verification

The solution produced by the model was checked and satisfies all the constraints of the problem. Specifically:

CONSTRAINT 1: Number of Two-Hour Sessions per Course and Section

- All courses have exactly the required number of two-hour sessions per week.
- The number of occurrences of each course matches its assigned requirement (e.g., Biology = 3 two-hour sessions per section, Math1 = 4 two-hour sessions in Section 1, etc.).

CONSTRAINT 2: At Most One Lesson per Time Slot and Section

- Each time slot (per day and section) contains at most one lesson. There are no overlaps.

CONSTRAINT 3: Each Teacher Teaches at Most One Section per Time Slot

- No teacher is assigned simultaneously to both sections in the same time slot. For example, Ms. Kirkofidou (Physics) has no double assignments at the same time.

CONSTRAINT 4: Physical Education Only Thursday 14:00–16:00 (Slot 3)

- PE1 and PE2 lessons are strictly scheduled on Thursday in Slot 3.

CONSTRAINT 5: Mr. Lathopraxis Does Not Teach Monday Morning

- The course Math2 is not assigned to Monday Slot 1 (08:00–10:00).

CONSTRAINT 6: Ms. Insulina Does Not Work on Wednesday

- Biology (taught by Insulina) does not appear anywhere on Wednesday.

CONSTRAINT 7: No More Than One Session of the Same Course per Day

- There are no instances where the same course is repeated twice on the same day for the same section.

CONSTRAINT 8: First Monday Slot is Reserved for Study

- No lesson is scheduled in Slot 1 on Monday (08:00–10:00) for either section.

Conclusion

The solution is **feasible, optimally structured, and fully compliant with all problem constraints**. The presented timetable tables can be used as a valid weekly schedule for both sections.

QUESTION 2

Warehouse Location Optimization Problem Modeling

Description

We want to select which warehouses from a set of available locations will operate in order to serve the sales centers at **minimum total cost**. The total cost includes:

- The **fixed operating cost** of each warehouse.
- The **transportation cost** to satisfy the demand of each sales center.

Sets

- $I = \{1, 2, \ldots, 12\}$: Set of available warehouses
- $J = \{1, 2, \ldots, 12\}$: Set of sales centers

Parameters

- f_i: Fixed cost of operating warehouse i
- c_{ij}: Transportation cost to fully satisfy the demand of center j from warehouse i
- d_j: Demand of sales center j (in tons)
- cap_i: Capacity of warehouse i (in tons)
- If $c_{ij} = \infty$, then warehouse i **cannot serve** center j

---

Decision Variables

- $y_i \in \{0, 1\}$:
  - $y_i = 1$ if warehouse i is open, 0 otherwise
- $x_{ij} \in [0, 1]$:
  - Fraction of the demand of center j served by warehouse i

Objective Function

Minimize the total fixed and transportation costs:

$$\min \sum_{i \in I} f_i \cdot y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} \cdot x_{ij}$$

Constraints

1. Full demand coverage for each center:

$$\sum_{i \in I} x_{ij} = 1 \quad \forall j \in J$$

---

2. Shipment from warehouse i cannot exceed its capacity (if open):

$$\sum_{j \in J} x_{ij} \cdot d_j \leq cap_i \cdot y_i \quad \forall i \in I$$

---

3. No shipment from warehouse i to center j if $c_{ij} = \infty$:

$$x_{ij} = 0 \quad \text{if } c_{ij} = \infty$$

---

4. Variable definitions:

- $y_i \in \{0, 1\}$
- $x_{ij} \in [0, 1]$

Gurobi code

In [7]:

```
# Sets of warehouses and sales centers
I = list(range(12))  # warehouses
J = list(range(12))  # sales centers

# Fixed cost per warehouse (in thousand €)
f = [3500, 9000, 10000, 4000, 3000, 9000, 9000, 3000, 4000, 10000, 9000, 350

# Capacity per warehouse (in tons)
cap = [300, 250, 100, 180, 275, 300, 200, 220, 270, 250, 230, 180]

# Demand per sales center (in tons)
d = [120, 80, 75, 100, 110, 100, 90, 60, 30, 150, 95, 120]

# Transportation costs (in thousand €)
```

```python
inf = 1e6  # representation of infinity
c_raw = [
    [100, 80, 50, 50, 60, 100, 120, 90, 60, 70, 65, 110],
    [120, 90, 60, 70, 65, 110, 140, 110, 80, 80, 75, 130],
    [140, 110, 80, 80, 75, 130, 160, 125, 100, 100, 80, 150],
    [160, 125, 100, 100, 80, 150, 190, 150, 130, inf, inf, inf],
    [190, 150, 130, inf, inf, inf, 180, 150, 50, 50, 60, 100],
    [200, 180, 150, inf, inf, inf, 100, 120, 90, 60, 75, 110],
    [120, 90, 60, 70, 65, 110, 140, 110, 80, 80, 75, 130],
    [120, 90, 60, 70, 65, 110, 140, 110, 80, 80, 75, 130],
    [140, 110, 80, 80, 75, 130, 160, 125, 100, 100, 80, 150],
    [160, 125, 100, 100, 80, 150, 190, 150, 130, inf, inf, inf],
    [190, 150, 130, inf, inf, inf, 200, 180, 150, inf, inf, inf],
    [200, 180, 150, inf, inf, inf, 100, 80, 50, 50, 60, 100]
]
c = np.array([[cost if cost != inf else inf for cost in row] for row in c_ra

# Create model
model = gp.Model("FacilityLocation")

# Solver parameters for high accuracy
model.setParam("MIPGap", 0.0)
model.setParam("MIPGapAbs", 0.0)
model.setParam("TimeLimit", 300)   # 5 minutes
model.setParam("Presolve", 2)
model.setParam("Heuristics", 0.1)
model.setParam("Cuts", 2)

# Decision variables
x = model.addVars(I, J, vtype=GRB.CONTINUOUS, lb=0, ub=1, name="x")
y = model.addVars(I, vtype=GRB.BINARY, name="y")

# Objective function: minimize total fixed + transportation cost
model.setObjective(
    gp.quicksum(f[i] * y[i] for i in I) +
    gp.quicksum(c[i][j] * x[i, j] for i in I for j in J),
    GRB.MINIMIZE
)

# Constraints: full demand coverage
for j in J:
    model.addConstr(gp.quicksum(x[i, j] for i in I) == 1)

# Constraints: warehouse capacities
for i in I:
    model.addConstr(gp.quicksum(x[i, j] * d[j] for j in J) <= cap[i] * y[i])

# Constraints: forbidden assignments (cost = inf)
for i in I:
    for j in J:
        if c[i][j] >= inf:
            model.addConstr(x[i, j] == 0)

# Solve
model.optimize()
```

```python
# Print results
print("\nWarehouses to be opened:")
for i in I:
    if y[i].X > 0.5:
        print(f"Warehouse {i+1}")

print("\nAssignments:")
for i in I:
    for j in J:
        if x[i, j].X > 1e-6:
            quantity = x[i, j].X * d[j]
            print(f"Center {j+1} served {x[i,j].X:.2f} from Warehouse {i+1}

# Display summary tables if optimal
if model.Status == GRB.OPTIMAL:
    print(f"\nTotal Cost: {round(model.ObjVal, 2)} thousand €")

    # List of active warehouses
    active_warehouses = [i for i in I if y[i].X > 0.5]

    # Compute total load per warehouse
    warehouse_loads = {i: 0.0 for i in active_warehouses}
    for i in active_warehouses:
        for j in J:
            if x[i, j].X > 1e-6:
                warehouse_loads[i] += x[i, j].X * d[j]

    # Table 1: active warehouses and total quantity served
    df1 = pd.DataFrame({
        'Warehouse': [i+1 for i in active_warehouses],
        'Total Tons Served': [round(warehouse_loads[i], 1) for i in active_w
    })
    display(df1)

    # Table 2: which centers are served by which warehouses
    assignments = []
    for i in active_warehouses:
        for j in J:
            if x[i, j].X > 1e-6:
                assignments.append({
                    'Center': j + 1,
                    'Warehouse': i + 1,
                    'Fraction of Demand': round(x[i, j].X, 2),
                    'Tons': round(x[i, j].X * d[j], 1)
                })

    df2 = pd.DataFrame(assignments)
    display(df2)
```

```
Set parameter MIPGap to value 0
Set parameter MIPGapAbs to value 0
Set parameter TimeLimit to value 300
Set parameter Presolve to value 2
Set parameter Heuristics to value 0.1
Set parameter Cuts to value 2
Gurobi Optimizer version 12.0.3 build v12.0.3rc0 (linux64 - "Ubuntu 22.04.4
LTS")

CPU model: Intel(R) Xeon(R) CPU @ 2.20GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 1 physical cores, 2 logical processors, using up to 2 threads

Non-default parameters:
TimeLimit  300
MIPGap  0
MIPGapAbs  0
Heuristics  0.1
Cuts  2
Presolve  2

Optimize a model with 45 rows, 156 columns and 321 nonzeros
Model fingerprint: 0xd36b6e96
Variable types: 144 continuous, 12 integer (12 binary)
Coefficient statistics:
  Matrix range     [1e+00, 3e+02]
  Objective range  [5e+01, 1e+06]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 42695.000000
Presolve removed 21 rows and 21 columns
Presolve time: 0.00s
Presolved: 24 rows, 135 columns, 258 nonzeros
Variable types: 123 continuous, 12 integer (12 binary)
Root relaxation presolved: 24 rows, 135 columns, 258 nonzeros


Root relaxation: objective 1.574011e+04, 33 iterations, 0.00 seconds (0.00 w
ork units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 15740.1111    0    1 42695.0000 15740.1111  63.1%     -    0s
H    0     0                    17976.222222 15740.1111  12.4%     -    0s
     0     0 15845.2485    0    2 17976.2222 15845.2485  11.9%     -    0s
     0     0 15845.2485    0    1 17976.2222 15845.2485  11.9%     -    0s
     0     0 16295.4271    0    2 17976.2222 16295.4271  9.35%     -    0s
     0     0 16804.6667    0    3 17976.2222 16804.6667  6.52%     -    0s
     0     0 17929.2315    0    2 17976.2222 17929.2315  0.26%     -    0s
H    0     0                    17930.369484 17929.2315  0.01%     -    0s
*    0     0               0    17929.231459 17929.2315  0.00%     -    0s

Cutting planes:
  Gomory: 2
  Cover: 1
  Implied bound: 2
```

```
  MIR: 2
  Flow cover: 5

Explored 1 nodes (79 simplex iterations) in 0.06 seconds (0.00 work units)
Thread count was 2 (of 2 available processors)

Solution count 2: 17929.2 42695

Optimal solution found (tolerance 0.00e+00)
Best objective 1.792923145933e+04, best bound 1.792923145933e+04, gap 0.000
0%

Warehouses to be opened:
Warehouse 1
Warehouse 5
Warehouse 8
Warehouse 9
Warehouse 12

Assignments:
Center 1 served 1.00 from Warehouse 1 (120.0 tons)
Center 2 served 0.06 from Warehouse 1 (5.0 tons)
Center 3 served 1.00 from Warehouse 1 (75.0 tons)
Center 4 served 1.00 from Warehouse 1 (100.0 tons)
Center 9 served 1.00 from Warehouse 5 (30.0 tons)
Center 10 served 0.83 from Warehouse 5 (125.0 tons)
Center 12 served 1.00 from Warehouse 5 (120.0 tons)
Center 2 served 0.94 from Warehouse 8 (75.0 tons)
Center 5 served 0.41 from Warehouse 8 (45.0 tons)
Center 6 served 1.00 from Warehouse 8 (100.0 tons)
Center 5 served 0.59 from Warehouse 9 (65.0 tons)
Center 11 served 0.95 from Warehouse 9 (90.0 tons)
Center 7 served 1.00 from Warehouse 12 (90.0 tons)
Center 8 served 1.00 from Warehouse 12 (60.0 tons)
Center 10 served 0.17 from Warehouse 12 (25.0 tons)
Center 11 served 0.05 from Warehouse 12 (5.0 tons)

Total Cost: 17929.23 thousand €
```

	Warehouse	Total Tons Served
0	1	300.0
1	5	275.0
2	8	220.0
3	9	155.0
4	12	180.0

	Center	Warehouse	Fraction of Demand	Tons
0	1	1	1.00	120.0
1	2	1	0.06	5.0
2	3	1	1.00	75.0
3	4	1	1.00	100.0
4	9	5	1.00	30.0
5	10	5	0.83	125.0
6	12	5	1.00	120.0
7	2	8	0.94	75.0
8	5	8	0.41	45.0
9	6	8	1.00	100.0
10	5	9	0.59	65.0
11	11	9	0.95	90.0
12	7	12	1.00	90.0
13	8	12	1.00	60.0
14	10	12	0.17	25.0
15	11	12	0.05	5.0

Solution Verification

1. **Full demand coverage per center:**
   Each center has total coverage equal to 100% (assignment fractions sum to 1.00).

2. **Respecting warehouse capacities:**
   Total quantities shipped from each active warehouse are:

   - Warehouse 1: 120 + 5 + 75 + 100 = 300 tons ≤ 300
   - Warehouse 5: 30 + 125 + 120 = 275 tons ≤ 275
   - Warehouse 8: 75 + 45 + 100 = 220 tons ≤ 220
   - Warehouse 9: 65 + 90 = 155 tons < 270
   - Warehouse 12: 90 + 60 + 25 + 5 = 180 tons ≤ 180
3. **No assignments from forbidden routes:**
   There are no assignments in x_{ij} for cases where the cost $c_{ij} = \infty$.

Conclusion

The solution produced by the model is **feasible**, **satisfies all problem constraints**, and provides the **optimal set of warehouses and assignments** while minimizing total cost.