Mandatory Assignment

Torger Bocianowski

Task 1

Sets

S = suppliers

P = plants

Parameters

Sup_s = the supply for a supplier s.

y_p = the yield per unit at a plant p.

O_p = constant opening cost of a plant p.

u_{sp} = unit cost from supplier s to plant p.

d_{sp} = distance driven (km) from supplier s to plant p by truck.

C_t = truck loading / unloading cost ($\$10,000$).

$CapT$ = truck capacity.

$CapP_p$ = capacity at plant p.

Decision Variables

$b_p \in \{0,1\}$ = binary variable, whether a plant at location p should be built or not.

x_{sp} = tons of biomass transported from supplier s to plant p.

$t_{sp} \in \mathbb{Z}$ = integer variable, the number of trucks used from supplier to plant.

Formulating the Objective Function & Constraints

minimize Cost: $\sum_{p \in P} O_p b_p$

$$+ \sum_{s \in S} \sum_{p \in P} (u_{sp} x_{sp} d_{sp} + C_t t_{sp})$$

subject to:	$\sum_{p \in P} x_{sp} \cdot b_p \leq Sup_s$	$\forall s \in S$	(Supply)
	$\sum_{s \in S} \sum_{p \in P} x_{sp} \cdot y_p \geq 500,000,000$		(Demand)
	$x_{sp} \leq t_{sp} \cdot CapT_{sp}$	$\forall s \in S, \forall p \in P$	(Number of trucks)
	$\sum_{s \in S} x_{sp} \leq CapP_p b_p$	$\forall p \in P$	(Plant capacity)

Programming

```python
import pandas as pd
import gurobipy as gp
from gurobipy import GRB

gp.setParam("OutputFlag", 0)

# Dataframes
suppliers_df = pd.read_csv("data/suppliers.csv")
plants_df = pd.read_csv("data/plants.csv")
roads_df = pd.read_csv("data/roads_s_p.csv")
```

Sets

```python
suppliers = suppliers_df.set_index("supplier")
suppliers = suppliers.to_dict(orient="index")

plants = plants_df.set_index("plant")
plants = plants.to_dict(orient="index")
```

Defining the Model

```python
m = gp.Model("SupplyChain")
```

Parameters

```python
# Helper function
def sp_param(df: pd.DataFrame, param: str) -> dict:
    df = df[["supplier", "plant", param]]
    df = df.set_index(["supplier", "plant"])
    df = df.to_dict(orient="index")
    return {(s, p): df[s, p][param] for s in suppliers for p in plants}

dist_s_p = sp_param(roads_df, "dist_s_p")
unit_cost_s_p = sp_param(roads_df, "cost_per_unit_s_p")
truck_cost_s_p = sp_param(roads_df, "truck_cost_s_p")
truck_cap_s_p = sp_param(roads_df, "truck_cap_s_p")
```

Decision Variables

```python
build = m.addVars(plants, vtype=GRB.BINARY, name="build")
biomass = m.addVars(suppliers, plants, name="biomass")
num_trucks = m.addVars(suppliers, plants, vtype=GRB.INTEGER, name="trucks")
```

Objective Function

```python
m.setObjective(
    gp.quicksum(
        plants[p]["plant_cost"] * build[p]
        for p in plants
    )
    + gp.quicksum(
        unit_cost_s_p[s, p] * biomass[s, p] * dist_s_p[s, p]
                        + (truck_cost_s_p[s, p] * num_trucks[s, p])
        for s in suppliers
        for p in plants
    ),
    GRB.MINIMIZE,
)
```

Constraints

```python
DEMAND = 500_000_000

m.addConstrs(
    (
        gp.quicksum(
            biomass[s, p] for p in plants
        ) <= suppliers[s]["supply"]
        for s in suppliers
    ),
    name="supply",
)

m.addConstr(
    (
        gp.quicksum(
            biomass[s, p] * plants[p]["yield_per_unit"]
                    for s in suppliers
                    for p in plants
        ) >= DEMAND
    ),
    name="demand",
)

m.addConstrs(
    (
        gp.quicksum(
            biomass[s, p] for s in suppliers
        ) <= plants[p]["plant_cap"] * build[p]
        for p in plants
    ),
    name="plant_cap",
)

m.addConstrs(
    (
        biomass[s, p] <= num_trucks[s, p] * truck_cap_s_p[s, p]
        for s in suppliers
        for p in plants
    ),
    name="num_trucks",
)

gp.setParam("OutputFlag", 0)
```

Solving the Model

```python
m.optimize()
```

```python
locations = [p for p in plants if build[p].x == 1]
total_biomass = sum(biomass[s, p].x for s in suppliers for p in plants)
total_num_trucks = sum(num_trucks[s, p].x for s in suppliers for p in plants)

print("RESULTS\n")
print(f"({len(locations)}) Plants: {locations[:6]}")
print(f"\t     {locations[6:]}\n")
print(f"Total biomass: {total_biomass:,.0f} Mg")
print(f"Trucks needed: {total_num_trucks:.0f}")
print(f"\nTotal cost: ${m.objVal:,.0f}")
```

```
RESULTS

(11) Plants: [541, 9063, 9076, 9102, 9107, 9155]
             [9178, 9183, 9203, 10058, 10066]

Total biomass: 2,155,172 Mg
Trucks needed: 4372

Total cost: $8,501,427,753
```

Interpreting the Results

As shown above,

- there are **11 locations** at which we want to **build plants**,
- we need **4374 trucks** to transport **~2.155Mg** biomass and
- the **total cost** is **~8.5 billion** dollars

Task 2

Sets

S = suppliers

H = hubs

P = plants

Parameters

Sup_s = the supply for a supplier s.

y_p = the yield per unit at a plant p.

O_h = constant opening cost of a hub h.

O_p = constant opening cost of a plant p.

u_{sh} = unit cost from supplier s to hub h.

u_{hp} = unit cost from hub h to plant p.

d_{sh} = distance driven (km) from supplier s to hub h by truck.

d_{hp} = distance driven (km) from hub h to plant p by train.

C_t = truck loading / unloading cost ($\$10,000$).

C_r = train loading / unloading cost ($\$60,000$).

$CapT$ = truck capacity.

$CapR$ = train capacity.

$CapH_h$ = capacity at hub h.

$CapP_p$ = capacity at plant p.

Decision Variables

$b_h \in \{0,1\}$ = binary variable, whether a hub at location h should be built or not.

$b_p \in \{0,1\}$ = binary variable, whether a plant at location p should be built or not.

x_{sh} = tons of biomass transported from supplier s to hub h.

x_{hp} = tons of biomass transported from hub h to plant p.

$t_{sh} \in \mathbb{Z}$ = integer variable, the number of trucks to be used from supplier to hub.

$r_{hp} \in \mathbb{Z}$ = integer variable, the number of trains to be used from hub to plant.

Formulating the Objective Function & Constraints

minimize Cost: $\sum_{h \in H} O_h \cdot b_h$

$$+ \sum_{p \in P} O_p \cdot b_p$$

$$+ \sum_{s \in S} \sum_{h \in H} (u_{sh} x_{sh} d_{sh}) + C_t t_{sh}$$

$$+ \sum_{h \in H} \sum_{p \in P} (u_{hp} x_{hp} d_{hp}) + C_r r_{hp}$$

subject to:	$\sum_{h \in H} x_{sh} \cdot b_h \leq Sup_s$	$\forall s \in S$	(Supply)
	$\sum_{h \in H} \sum_{p \in P} x_{hp} \cdot y_p \geq 500,000,000$		(Demand)
	$x_{sh} \leq t_{sh} \cdot CapT_{sh}$	$\forall s \in S, \forall p \in P$	(Number of trucks)
	$x_{hp} \leq r_{hp} \cdot CapR_{hp}$	$\forall h \in H, \forall p \in P$	(Number of trains)
	$\sum_{s \in S} x_{sh} \leq CapH_h b_h$	$\forall h \in H$	(Hub capacity)
	$\sum_{h \in H} x_{hp} \leq CapP_p b_p$	$\forall p \in P$	(Plant capacity)
	$\sum_{s \in S} x_{sh} = \sum_{p \in P} x_{hp}$	$\forall h \in H$	(Flow balance)

Programming

Sets

We add a new set hubs. The sets suppliers and plants remain the same from the previous task.

```python
# Dataframes
hubs_df = pd.read_csv("data/hubs.csv")
railroads_df = pd.read_csv("data/railroads_h_p.csv")
roads_df = pd.read_csv("data/roads_s_h.csv")

# Sets
hubs = hubs_df.set_index("hub")
hubs = hubs.to_dict(orient="index")
```

Defining the Model

```python
m = gp.Model("SupplyChain2")
```

Parameters

```python
# Helper functions
def sh_param(df: pd.DataFrame, param: str) -> dict:
    df = df[["supplier", "hub", param]]
    df = df.set_index(["supplier", "hub"])
    df = df.to_dict(orient="index")
    return {(s, h): df[s, h][param] for s in suppliers for h in hubs}

def hp_param(df: pd.DataFrame, param: str) -> dict:
    df = df[["hub", "plant", param]]
    df = df.set_index(["hub", "plant"])
    df = df.to_dict(orient="index")
    return {(h, p): df[h, p][param] for h in hubs for p in plants}

# params from roads_s_h.csv
dist_s_h = sh_param(roads_df, "dist_s_h")
unit_cost_s_h = sh_param(roads_df, "cost_per_unit_s_h")
truck_cost_s_h = sh_param(roads_df, "truck_cost_s_h")
truck_cap_s_h = sh_param(roads_df, "truck_cap_s_h")

# params from railroads_h_p.csv
dist_h_p = hp_param(railroads_df, "dist_h_p")
unit_cost_h_p = hp_param(railroads_df, "cost_per_unit_h_p")
train_cost_h_p = hp_param(railroads_df, "train_cost_h_p")
train_cap_h_p = hp_param(railroads_df, "train_cap_h_p")
```

Decision Variables

```python
build_hub = m.addVars(hubs, vtype=GRB.BINARY, name="build_hub")
build_plant = m.addVars(plants, vtype=GRB.BINARY, name="build_plant")

biomass_s_h = m.addVars(suppliers, hubs, name="biomass_s_h")
biomass_h_p = m.addVars(hubs, plants, name="biomass_h_p")

num_trucks = m.addVars(suppliers, hubs, vtype=GRB.INTEGER, name="trucks")
num_trains = m.addVars(hubs, plants, vtype=GRB.INTEGER, name="trains")
```

Objective Function

```python
m.setObjective(
    gp.quicksum(
        hubs[h]["hub_cost"] * build_hub[h]
        for h in hubs
    )
    + gp.quicksum(
        plants[p]["plant_cost"] * build_plant[p]
        for p in plants
    )
    + gp.quicksum(
        unit_cost_s_h[s, h] * biomass_s_h[s, h] * dist_s_h[s, h]
                        + (truck_cost_s_h[s, h] * num_trucks[s, h])
        for s in suppliers
        for h in hubs
    )
    + gp.quicksum(
        unit_cost_h_p[h, p] * biomass_h_p[h, p] * dist_h_p[h, p]
                        + (train_cost_h_p[h, p] * num_trains[h, p])
        for h in hubs
        for p in plants
    ),
    GRB.MINIMIZE,
)
```

Constraints

```python
DEMAND = 500_000_000

m.addConstrs(
    (
        gp.quicksum(
            biomass_s_h[s, h] * build_hub[h] for h in hubs
        ) <= suppliers[s]["supply"]
        for s in suppliers
    ),
    name="supply",
)

m.addConstr(
    (
        gp.quicksum(
            biomass_h_p[h, p] * plants[p]["yield_per_unit"]
                    for h in hubs
                    for p in plants
        ) >= DEMAND
    ),
    name="demand",
)

m.addConstrs(
    (
        biomass_s_h[s, h] <= num_trucks[s, h] * truck_cap_s_h[s, h]
        for s in suppliers
        for h in hubs
    ),
    name="num_trucks",
)

m.addConstrs(
    (
        biomass_h_p[h, p] <= num_trains[h, p] * train_cap_h_p[h, p]
        for h in hubs
        for p in plants
    ),
    name="num_trains",
)

m.addConstrs(
    (
        gp.quicksum(
            biomass_s_h[s, h] for s in suppliers
        ) <= hubs[h]["hub_cap"] * build_hub[h]
        for h in hubs
    ),
    name="hub_capacity",
)


m.addConstrs(
    (
        gp.quicksum(
            biomass_h_p[h, p] for h in hubs
        ) <= plants[p]["plant_cap"] * build_plant[p]
        for p in plants
    ),
    name="plant_capacity",
)

m.addConstrs(
    (
        gp.quicksum(biomass_s_h[s, h] for s in suppliers)
        == gp.quicksum(biomass_h_p[h, p] for p in plants)
        for h in hubs
    ),
    name="balance_flow",
)

gp.setParam("OutputFlag", 0)
```

Solving the Model

```python
m.optimize()
```

```python
plants_locations = [p for p in plants if build_plant[p].x == 1]
hubs_locations = [h for h in hubs if build_hub[h].x == 1]

total_biomass_s_h = sum(biomass_s_h[s, h].x for s in suppliers for h in hubs)
total_biomass_h_p = sum(biomass_h_p[h, p].x for h in hubs for p in plants)

total_num_trucks = sum(num_trucks[s, h].x for s in suppliers for h in hubs)
total_num_trains = sum(num_trains[h, p].x for h in hubs for p in plants)

print("RESULTS\n")
print(f" ({len(plants_locations)}) Plants: {plants_locations}")
print(f"({len(hubs_locations)})   Hubs: {hubs_locations[:10]}")
print(f"\t     {hubs_locations[10:20]}")
print(f"\t     {hubs_locations[20:]}\n")
print(f"Total biomass: {total_biomass_s_h:,.0f} Mg") # biomass_s_h = biomass_h_p

print(f"Trucks needed: {total_num_trucks:.0f}")
print(f"Trains needed: {total_num_trains:.0f}")
print(f"\nTotal cost: ${m.objVal:,.0f}")
```

```
RESULTS

 (8) Plants: [541, 9047, 9060, 9091, 9178, 9183, 9203, 10066]
(31)   Hubs: [17201, 17218, 17359, 17372, 17395, 17404, 17447, 17466, 17507, 17592]
             [17620, 17679, 17717, 17784, 17792, 17822, 17829, 17896, 17931, 17934]
             [17942, 17943, 18029, 18042, 18063, 18082, 18127, 18286, 18288, 18294, 18303]

Total biomass: 2,155,172 Mg
Trucks needed: 4372
Trains needed: 124

Total cost: $5,135,420,807
```

Interpreting the Results

As shown above,

- there are **8 locations** at which we want to **build plants**,
- there are **31 locations** at which we want to **build hubs**,
- we need **4372 trucks** and **124** trains to transport **~2.155Mg** biomass and,
- the **total cost** is **~5.135 billion** dollars

Task 3

Sets

S = suppliers

H = hubs

P = plants

T = third party suppliers

Parameters

Sup_s = the supply for a supplier s.

y_p = the yield per unit at a plant p.

O_h = constant opening cost of a hub h.

O_p = constant opening cost of a plant p.

u_{sh} = unit cost from supplier s to hub h.

u_{hp} = unit cost from hub h to plant p.

u_{th} = unit cost from t to h ($\$2,000$).

d_{sh} = distance driven (km) from supplier s to hub h by truck.

d_{hp} = distance driven (km) from hub h to plant p by train.

C_t = truck loading / unloading cost ($\$10,000$).

C_r = train loading / unloading cost ($\$60,000$).

$CapT$ = truck capacity.

$CapR$ = train capacity.

$CapH_h$ = capacity at hub h.

$CapP_p$ = capacity at plant p.

Decision Variables

$b_h \in \{0,1\}$ = binary variable, whether a hub at location h should be built or not.

$b_p \in \{0,1\}$ = binary variable, whether a plant at location p should be built or not.

x_{sh} = tons of biomass transported from supplier s to hub h.

x_{hp} = tons of biomass transported from hub h to plant p.

x_{th} = tons of biomass transported from third party location t to hub h.

$t_{sh} \in \mathbb{Z}$ = integer variable, the number of trucks to be used from supplier to hub.

$r_{hp} \in \mathbb{Z}$ = integer variable, the number of trains to be used from hub to plant.

Formulating the Objective Function & Constraints

minimize Cost: $\sum_{h \in H} O_h \cdot b_h$

$$+ \sum_{p \in P} O_p \cdot b_p$$

$$+ \sum_{s \in S} \sum_{h \in H} (u_{sh} x_{sh} d_{sh}) + C_t t_{sh}$$

$$+ \sum_{h \in H} \sum_{p \in P} (u_{hp} x_{hp} d_{hp}) + C_r r_{hp}$$

$$+ \sum_{t \in T} \sum_{h \in H} (u_{th} x_{th})$$

subject to:	$\sum_{h \in H} x_{sh} \cdot b_h \leq Sup_s$	$\forall s \in S$	(Supply)
	$\sum_{h \in H} \sum_{p \in P} x_{hp} \cdot y_p \geq 800,000,000$		(Demand)
	$x_{sh} \leq t_{sh} \cdot CapT_{sh}$	$\forall s \in S, \forall h \in H$	(Number of trucks)
	$x_{hp} \leq r_{hp} \cdot CapR_{hp}$	$\forall h \in H, \forall p \in P$	(Number of trains)
	$\sum_{s \in S} x_{sh} + \sum_{t \in T} x_{th} \leq CapH_h b_h$	$\forall h \in H$	(Hub capacity)
	$\sum_{h \in H} x_{hp} \leq CapP_p b_p$	$\forall p \in P$	(Plant capacity)
	$\sum_{s \in S} x_{sh} + \sum_{t \in T} x_{th} = \sum_{p \in P} x_{hp}$	$\forall h \in H$	(Flow balance)

Programming

Sets

Adding a set (in the same format) for the third party that emulates infinite supply.

```python
third_party_supplier = {
    "x": {
        "supply": float("inf"),
    }
}
```

Defining the Model

```python
m = gp.Model("SupplyChain3")
```

Parameters

The parameters stay the same from task 2, but we need to add the unit cost of $\$2000/Mg$

```python
unit_cost_third_party = 2000
```

Decision Variables

We add a new decision variable for the biomass from the third party supplier.

```python
build_plant = m.addVars(plants, vtype=GRB.BINARY, name="build_plant")
build_hub = m.addVars(hubs, vtype=GRB.BINARY, name="build_hub")

num_trucks = m.addVars(suppliers, hubs, vtype=GRB.INTEGER, name="trucks")
num_trains = m.addVars(hubs, plants, vtype=GRB.INTEGER, name="trains")

biomass_s_h = m.addVars(suppliers, hubs, name="biomass_s_h")
biomass_h_p = m.addVars(hubs, plants, name="biomass_h_p")

third_party_biomass = m.addVars(third_party_supplier, hubs, name="third_party_biomass")
```

Objective Function

```python
m.setObjective(
    gp.quicksum(
        hubs[h]["hub_cost"] * build_hub[h]
        for h in hubs
    )
    + gp.quicksum(
        plants[p]["plant_cost"] * build_plant[p]
        for p in plants
    )
    + gp.quicksum(
        unit_cost_s_h[s, h] * biomass_s_h[s, h] * dist_s_h[s, h]
                        + (truck_cost_s_h[s, h] * num_trucks[s, h])
        for s in suppliers
        for h in hubs
    )
    + gp.quicksum(
        unit_cost_h_p[h, p] * biomass_h_p[h, p] * dist_h_p[h, p]
                        + (train_cost_h_p[h, p] * num_trains[h, p])
        for h in hubs
        for p in plants
    )
    + gp.quicksum(
        unit_cost_third_party * third_party_biomass[t, h]
        for t in third_party_supplier
        for h in hubs
    ),
    GRB.MINIMIZE,
)
```

Constraints

The demand is now $800,000,000$. We must also alter the constraint for the hub capacity and the flow balance constraint.

```python
DEMAND = 800_000_000

m.addConstrs(
    (
        gp.quicksum(
            biomass_s_h[s, h] * build_hub[h] for h in hubs
        ) <= suppliers[s]["supply"]
        for s in suppliers
    ),
    name="supply",
)

m.addConstr(
    gp.quicksum(
        biomass_h_p[h, p] * plants[p]["yield_per_unit"]
                for h in hubs
                for p in plants
        ) >= DEMAND,
    name="demand",
)

m.addConstrs(
    (
        biomass_s_h[s, h] <= num_trucks[s, h] * truck_cap_s_h[s, h]
        for s in suppliers
        for h in hubs
    ),
    name="num_trucks",
)

m.addConstrs(
    (
        biomass_h_p[h, p] <= num_trains[h, p] * train_cap_h_p[h, p]
        for h in hubs
        for p in plants
    ),
    name="num_trains",
)

m.addConstrs(
    (
        gp.quicksum(
            biomass_s_h[s, h] for s in suppliers
        )
        + gp.quicksum(
            third_party_biomass[t, h] for t in third_party_supplier
        ) <= hubs[h]["hub_cap"] * build_hub[h]
        for h in hubs
    ),
    name="hub_capacity",
)

m.addConstrs(
    (
        gp.quicksum(
            biomass_h_p[h, p] for h in hubs
        ) <= plants[p]["plant_cap"] * build_plant[p]
        for p in plants
    ),
    name="plant_capacity",
)

m.addConstrs(
    (
        gp.quicksum(biomass_s_h[s, h] for s in suppliers)
        + gp.quicksum(third_party_biomass[t, h] for t in third_party_supplier)
        == gp.quicksum(biomass_h_p[h, p] for p in plants)
        for h in hubs
    ),
    name="balance_flow",
)

m.setParam("OutputFlag", 0)
```

Solving the Model

```python
m.optimize()
```

```python
plants_locations = [p for p in plants if build_plant[p].x == 1]
hubs_locations = [h for h in hubs if build_hub[h].x == 1]

total_biomass_s_h = sum(biomass_s_h[s, h].x for s in suppliers for h in hubs)
total_biomass_h_p = sum(biomass_h_p[h, p].x for h in hubs for p in plants)
total_biomass_third_party = sum(third_party_biomass[t, h].x for t in third_party_supplier for h in hubs)

total_num_trucks = sum(num_trucks[s, h].x for s in suppliers for h in hubs)
total_num_trains = sum(num_trains[h, p].x for h in hubs for p in plants)

print("RESULTS\n")
print(f" ({len(plants_locations)}) Plants: {plants_locations}")
print(f"({len(hubs_locations)})   Hubs: {hubs_locations[:9]}")
print(f"\t     {hubs_locations[9:]}")

print(f"\nTotal biomass sh:          {total_biomass_s_h:,.0f} Mg")
print(f"Total biomass third party: {total_biomass_third_party:,.0f} Mg")
print(f"Total biomass hp:          {total_biomass_h_p:,.0f} Mg")

print(f"\nTrucks needed: {total_num_trucks:.0f}")
print(f"Trains needed: {total_num_trains:.0f}")
print(f"\nTotal cost: ${m.objVal:,.0f}")
```

```
RESULTS

 (6) Plants: [9044, 9047, 9178, 9183, 9203, 10066]
(18)   Hubs: [17201, 17218, 17359, 17447, 17507, 17592, 17679, 17792, 17829]
             [17896, 17934, 17943, 18029, 18042, 18082, 18286, 18288, 18294]

Total biomass sh:          1,401,295 Mg
Total biomass third party: 2,046,981 Mg
Total biomass hp:          3,448,276 Mg

Trucks needed: 2833
Trains needed: 179

Total cost: $7,139,972,964
```

Interpreting the Results

As shown above,

- there are **6 locations** at which we want to **build plants**,
- there are **18 locations** at which we want to **build hubs**,

- we need **2833 trucks** and **179** trains to transport **~3.448Mg** biomass and,
- the **total cost** is **~7.14 billion** dollars