# Optimization

Task 3

Nathan Hefner

**A. Using the mathematical representations that you developed in Task 2, write a program in either Python or R to solve the optimization problem computationally.**

**1. Demonstrate that the solver provided a solution to the optimization problem.**

Below is the solution provided by the code.

```
Ship 34397.0 tons from CVG to Leipzig
Ship 19000.0 tons from CVG to Hyderabad
Ship 36000.0 tons from CVG to San Bernardino
Ship 44350.0 tons from AFW to Leipzig
Ship 6500.0 tons from Leipzig to Paris
Ship 300.0 tons from Leipzig to Cologne
Ship 180.0 tons from Leipzig to Hanover
Ship 9100.0 tons from Leipzig to Bangalore
Ship 90.0 tons from Leipzig to Cagliari
Ship 185.0 tons from Leipzig to Catania
Ship 800.0 tons from Leipzig to Milan
Ship 1700.0 tons from Leipzig to Rome
Ship 170.0 tons from Leipzig to Katowice
Ship 2800.0 tons from Leipzig to Barcelona
Ship 3700.0 tons from Leipzig to Madrid
Ship 190.0 tons from Leipzig to Mobile
Ship 175.0 tons from Leipzig to Anchorage
Ship 38.0 tons from Leipzig to Fairbanks
Ship 2400.0 tons from Leipzig to Phoenix
Ship 7200.0 tons from Leipzig to Los Angeles
Ship 100.0 tons from Leipzig to Ontario
Ship 1200.0 tons from Leipzig to Riverside
Ship 1100.0 tons from Leipzig to Sacramento
Ship 1900.0 tons from Leipzig to San Francisco
Ship 240.0 tons from Leipzig to Stockton
Ship 1500.0 tons from Leipzig to Denver
Ship 3400.0 tons from Leipzig to Miami
Ship 185.0 tons from Leipzig to Lakeland
Ship 1600.0 tons from Leipzig to Tampa
Ship 3000.0 tons from Leipzig to Atlanta
Ship 500.0 tons from Leipzig to Honolulu
Ship 16.0 tons from Leipzig to Kahului/Maui
Ship 63.0 tons from Leipzig to Kona
Ship 5100.0 tons from Leipzig to Chicago
Ship 172.0 tons from Leipzig to Rockford
Ship 200.0 tons from Leipzig to Fort Wayne
Ship 173.0 tons from Leipzig to South Bend
Ship 300.0 tons from Leipzig to Des Moines
Ship 290.0 tons from Leipzig to Wichita
Ship 550.0 tons from Leipzig to New Orleans
Ship 1700.0 tons from Leipzig to Minneapolis
Ship 975.0 tons from Leipzig to Kansas City
Ship 1200.0 tons from Leipzig to St. Louis
Ship 480.0 tons from Leipzig to Omaha
Ship 450.0 tons from Leipzig to Albuquerque
Ship 900.0 tons from Leipzig to Charlotte
Ship 290.0 tons from Leipzig to Toledo
Ship 150.0 tons from Leipzig to Wilmington
Ship 1200.0 tons from Leipzig to Portland
Ship 1000.0 tons from Leipzig to Pittsburgh
Ship 1100.0 tons from Leipzig to San Juan
Ship 650.0 tons from Leipzig to Nashville
Ship 975.0 tons from Leipzig to Austin
Ship 3300.0 tons from Leipzig to Dallas
Ship 3300.0 tons from Leipzig to Houston
Ship 1100.0 tons from Leipzig to San Antonio
Ship 600.0 tons from Leipzig to Richmond
Ship 2000.0 tons from Leipzig to Seattle/Tacoma
Ship 260.0 tons from Leipzig to Spokane
Ship 570.0 tons from Hyderabad to Coimbatore
Ship 4840.0 tons from Hyderabad to Mumbai
Ship 30.0 tons from Hyderabad to Castle Donington
Ship 540.0 tons from Hyderabad to Hartford
Ship 1300.0 tons from Hyderabad to Baltimore
Ship 100.0 tons from Hyderabad to Manchester
Ship 11200.0 tons from Hyderabad to New York
Ship 420.0 tons from Hyderabad to Allentown
Ship 340.0 tons from San Bernardino to Cologne
Ship 19000.0 tons from San Bernardino to Delhi
Ship 9960.0 tons from San Bernardino to Mumbai
Ship 6700.0 tons from San Bernardino to London
```

**B. Analyze the output of your model to demonstrate that the solution satisfies the requirements of the problem.**

The output gives 71 optimized shipping routes that show how many tons of products should be transported to and from specific cities.

**1. Demonstrate that the constraints of the optimization problem are satisfied.**

Below shows the code and constraints being satisfied

```
prob.solve()

for h in hubs:
    total = sum(x[h][f].varValue for f in focus_cities if x[h][f].varValue is not None)
    cap = hubs[h]["capacity"]
    status = "OK ✅" if total <= cap else "VIOLATION ⚠️"
    print(f"Hub {h}: {total:.2f} / {cap} ({status})")
```

```
Hub CVG: 89397.00 / 95650 (OK ✅)
Hub AFW: 44350.00 / 44350 (OK ✅)
```

```
for f in focus_cities:
    values = [x[h][f].varValue for h in hubs if x[h][f].varValue is not None]
    total_shipment_to_focus_city = sum(values) if values else 0

    cap = focus_cities[f]["capacity"]
    status = "OK ✅" if total_shipment_to_focus_city <= cap else "VIOLATION ⚠️"
    print(f"Focus city {f}: {total_shipment_to_focus_city:.2f} / {cap} ({status})")
```

```
Focus city Leipzig: 78747.00 / 85000 (OK ✅)
Focus city Hyderabad: 19000.00 / 19000 (OK ✅)
Focus city San Bernardino: 36000.00 / 36000 (OK ✅)
```

```
for city, demand in flattened_centers.items():
    values = [y[f][city].varValue for f in focus_cities if y[f][city].varValue is not None]
    total_shipment_to_center = sum(values) if values else 0

    if abs(total_shipment_to_center - demand) <= 1e-3:
        print(f"Center {city}: demand satisfied ✅ ({total_shipment_to_center:.2f} / {demand})")
    else:
        print(f"Center {city}: demand VIOLATION ⚠️ ({total_shipment_to_center:.2f} / {demand})")
```

Center Paris: demand satisfied ✅ (6500.00 / 6500)
Center Cologne: demand satisfied ✅ (640.00 / 640)
Center Hanover: demand satisfied ✅ (180.00 / 180)
Center Bangalore: demand satisfied ✅ (9100.00 / 9100)
Center Coimbatore: demand satisfied ✅ (570.00 / 570)
Center Delhi: demand satisfied ✅ (19000.00 / 19000)
Center Mumbai: demand satisfied ✅ (14800.00 / 14800)
Center Cagliari: demand satisfied ✅ (90.00 / 90)
Center Catania: demand satisfied ✅ (185.00 / 185)
Center Milan: demand satisfied ✅ (800.00 / 800)
Center Rome: demand satisfied ✅ (1700.00 / 1700)
Center Katowice: demand satisfied ✅ (170.00 / 170)
Center Barcelona: demand satisfied ✅ (2800.00 / 2800)
Center Madrid: demand satisfied ✅ (3700.00 / 3700)
Center Castle Donington: demand satisfied ✅ (30.00 / 30)
Center London: demand satisfied ✅ (6700.00 / 6700)
Center Mobile: demand satisfied ✅ (190.00 / 190)
Center Anchorage: demand satisfied ✅ (175.00 / 175)
Center Fairbanks: demand satisfied ✅ (38.00 / 38)
Center Phoenix: demand satisfied ✅ (2400.00 / 2400)
Center Los Angeles: demand satisfied ✅ (7200.00 / 7200)
Center Ontario: demand satisfied ✅ (100.00 / 100)
Center Riverside: demand satisfied ✅ (1200.00 / 1200)
Center Sacramento: demand satisfied ✅ (1100.00 / 1100)
Center San Francisco: demand satisfied ✅ (1900.00 / 1900)
Center Stockton: demand satisfied ✅ (240.00 / 240)
Center Denver: demand satisfied ✅ (1500.00 / 1500)
Center Hartford: demand satisfied ✅ (540.00 / 540)
Center Miami: demand satisfied ✅ (3400.00 / 3400)
Center Lakeland: demand satisfied ✅ (185.00 / 185)
Center Tampa: demand satisfied ✅ (1600.00 / 1600)
Center Atlanta: demand satisfied ✅ (3000.00 / 3000)
Center Honolulu: demand satisfied ✅ (500.00 / 500)
Center Kahului/Maui: demand satisfied ✅ (16.00 / 16)
Center Kona: demand satisfied ✅ (63.00 / 63)
Center Chicago: demand satisfied ✅ (5100.00 / 5100)
Center Rockford: demand satisfied ✅ (172.00 / 172)
Center Fort Wayne: demand satisfied ✅ (200.00 / 200)
Center South Bend: demand satisfied ✅ (173.00 / 173)
Center Des Moines: demand satisfied ✅ (300.00 / 300)
Center Wichita: demand satisfied ✅ (290.00 / 290)
Center New Orleans: demand satisfied ✅ (550.00 / 550)
Center Baltimore: demand satisfied ✅ (1300.00 / 1300)
Center Minneapolis: demand satisfied ✅ (1700.00 / 1700)
Center Kansas City: demand satisfied ✅ (975.00 / 975)
Center St. Louis: demand satisfied ✅ (1200.00 / 1200)
Center Omaha: demand satisfied ✅ (480.00 / 480)
Center Manchester: demand satisfied ✅ (100.00 / 100)
Center Albuquerque: demand satisfied ✅ (450.00 / 450)
Center New York: demand satisfied ✅ (11200.00 / 11200)
Center Charlotte: demand satisfied ✅ (900.00 / 900)
Center Toledo: demand satisfied ✅ (290.00 / 290)
Center Wilmington: demand satisfied ✅ (150.00 / 150)
Center Portland: demand satisfied ✅ (1200.00 / 1200)
Center Allentown: demand satisfied ✅ (420.00 / 420)
Center Pittsburgh: demand satisfied ✅ (1000.00 / 1000)
Center San Juan: demand satisfied ✅ (1100.00 / 1100)
Center Nashville: demand satisfied ✅ (650.00 / 650)
Center Austin: demand satisfied ✅ (975.00 / 975)
Center Dallas: demand satisfied ✅ (3300.00 / 3300)
Center Houston: demand satisfied ✅ (3300.00 / 3300)
Center San Antonio: demand satisfied ✅ (1100.00 / 1100)
Center Richmond: demand satisfied ✅ (600.00 / 600)
Center Seattle/Tacoma: demand satisfied ✅ (2000.00 / 2000)
Center Spokane: demand satisfied ✅ (260.00 / 260)

```
for f in focus_cities:
    total_flow_in = sum(x[h][f].varValue for h in hubs)
    total_flow_out = sum(y[f][city].varValue for city in flattened_centers)
    if total_flow_in == total_flow_out:
        print(f"Flow balance for focus city {f} satisfied: {total_flow_in} == {total_flow_out}")
    else:
        print(f"Flow balance for focus city {f} violated: {total_flow_in} != {total_flow_out}")
```

```
Flow balance for focus city Leipzig satisfied: 78747.0 == 78747.0
Flow balance for focus city Hyderabad satisfied: 19000.0 == 19000.0
Flow balance for focus city San Bernardino satisfied: 36000.0 == 36000.0
```

**2. Demonstrate that the solution includes decision variables, constraints, and the objective function.**

Below is the code that includes decision variables, constraints, and the objective function.

```
prob = LpProblem("Cargo_Optimization", LpMinimize)

x = LpVariable.dicts(
    "shipment_from_hub_to_focus",
    (hubs.keys(), focus_cities.keys()),
    lowBound=0,
    cat="Continuous"
)

y = LpVariable.dicts(
    "shipment_from_focus_to_center",
    (focus_cities.keys(), flattened_centers.keys()),
    lowBound=0,
    cat="Continuous"
)

prob += (
    lpSum(
        get_cost(h, f, costs) * x[h][f]
        for h in hubs
        for f in focus_cities
        if get_cost(h, f, costs) is not None and get_cost(h, f, costs) < DEFAULT_COST
    )
    +
    lpSum(
        get_cost(f, c, costs) * y[f][c]
        for f in focus_cities
        for c in flattened_centers
        if get_cost(f, c, costs) is not None and get_cost(f, c, costs) < DEFAULT_COST
    )
), "Total_Shipping_Cost"
```

```
for h in hubs:
    prob += lpSum([x[h][f] for f in focus_cities]) <= hubs[h]["capacity"], f"Hub_{h}_Capacity"

for f in focus_cities:
    prob += lpSum([x[h][f] for h in hubs]) <= focus_cities[f]["capacity"], f"FocusCity_{f}_Capacity"

for city, demand in flattened_centers.items():
    prob += lpSum([y[f][city] for f in focus_cities]) == demand, f"Demand_{city}"

for f in focus_cities:
    prob += lpSum([x[h][f] for h in hubs]) == lpSum([y[f][city] for city in flattened_centers]), f"Flow_Balance_{f}"

prob.solve()

if LpStatus[prob.status] == "Optimal":
    for h in hubs:
        for f in focus_cities:
            if x[h][f].varValue > 0:
                print(f"Ship {x[h][f].varValue} tons from {h} to {f}")
    for f in focus_cities:
        for city in flattened_centers:
            if y[f][city].varValue > 0:
                print(f"Ship {y[f][city].varValue} tons from {f} to {city}")
else:
    print("No optimal solution found")
```

**3. Explain why the solution that your code outputs matches your expected output.**

This is the expected outcome because the solution consists of shipping routes that are optimized for the amount of product shipped and the shortest path taken to reach the cities and hubs, along with the capacity constraint being fulfilled, ensuring that each center will meet its demand.

**C. Provide a reflection on how the development of your approach matched your expectations of the process.**

For the most part, the development matched my expectations. I knew we would input the data set, the constraints, and the objective function and apply some calculus to help with computations. The hardest part for me was actually inputting the data itself, as it came from a Word document. I first tried to reference the Word document itself, but to no avail. I then pondered trying to input all the data manually into the code. I then came to my senses, and since the data were in tables already, it was simple to copy and paste the data all into separate CSV files that the code could then reference and read.

Sources

No sources were used except WGU material