(a) Hub Selection Constraint:

$$\sum_{i=1}^{8} x_i = 2$$

This constraint ensures that exactly two hubs are selected (x_i is a binary variable representing whether area i is selected as a hub).

(b) If area i is not selected as a hub, it cannot serve any other areas:

$$\sum_{j=1}^{8} y_{ij} \leq 8\, x_i, \quad i = 1, \ldots, 8$$

This constraint ensures that if area i is not selected as a hub ($x_i = 0$), then it cannot serve any other area.

(c) Each area j must be served by exactly one hub, and area i can serve only if it is selected as a hub:

$$\begin{aligned} \sum_{i=1}^{8} y_{ij} &= 1, & j &= 1, \ldots, 8, \\ y_{ij} &\leq x_i, & i, j &= 1, \ldots, 8. \end{aligned}$$

This set of constraints ensures that each area j is served by exactly one hub and that area i can only serve if it is selected as a hub.

(d) The demand d_j at area j is the total traffic flux arriving at area j:

$$d_j = \sum_{i=1}^{8} f_{ij}, \quad j = 1, \ldots, 8.$$

The demand d_j is the total traffic flux arriving at area j from all other areas.

(e) The total demand D is the sum of demands from all areas:

$$D = \sum_{j=1}^{8} d_j$$

For each hub i, the total demand it serves must not exceed 60% of the total demand D:

$$\sum_{j=1}^{8} d_j\, y_{ij} \leq 0.6\, D, \quad i = 1, \ldots, 8.$$

This constraint ensures load balancing for each hub, limiting the proportion of total demand that each hub can serve.

(f) The transportation cost is measured by the journey time multiplied by the demand:

$$Z_1 = \sum_{i=1}^{8} \sum_{j=1}^{8} t_{ij}\, d_j\, y_{ij}.$$

The objective function Z_1 calculates the total transportation cost based on the journey times and the demands.

(g) Introduce a new variable U as the upper bound of all selected delivery times:

$$\min\ Z_2 = U$$

Subject to:

$$t_{ij}\, y_{ij} \leq U, \quad i, j = 1, \ldots, 8.$$

The objective function Z_2 minimizes the maximum delivery time for all selected paths, ensuring that the delivery time for each path does not exceed U.

In [1]:
```
import pandas as pd
import numpy as np
from pulp import LpProblem, LpMinimize, LpVariable, lpSum, LpBinary, PULP

# Journey times (in minutes)
journey_data = {
    'From': ['Neihu', 'Yuanshan', 'Taipei', 'Sanzhong', 'Wugu', 'Freeway
    'Neihu': [None, 7.39, 9.52, 12.69, 17.31, 169.33, 25.80, 45.43],
    'Yuanshan': [3.19, None, 1.85, 4.93, 9.87, None, 18.61, 31.94],
    'Taipei': [11.93, 8.26, None, 3.13, 7.46, 177.62, 16.25, 26.90],
    'Sanzhong': [13.98, 10.18, 1.33, None, 4.11, None, 12.97, 22.29],
    'Wugu': [15.11, 11.91, 3.30, 2.21, None, None, 8.30, 12.64],
    'Freeway Bureau': [21.35, 18.08, 8.04, 8.02, 5.10, None, None, None],
    'Linkou North': [22.24, 19.12, 9.97, 9.24, 6.82, 3.01, None, 6.07],
    'Linkou South': [30.01, 25.02, 13.83, 14.12, 11.63, 8.38, 8.62, None]
}

# Load data into a DataFrame and set index
journey_df = pd.DataFrame(journey_data)
journey_df.set_index('From', inplace=True)
regions = list(journey_df.index)
# Traffic flow data (vehicles per hour), converted to vehicles per minute
```

```
traffic_data = {
    'From': ['Neihu', 'Yuanshan', 'Taipei', 'Sanzhong', 'Wugu', 'Freeway
    'Neihu': [None, 132.016, 129.421, 65.829, 57.799, 0.005, 37.845, 0.00
    'Yuanshan': [131.652, None, 287.611, 170.747, 122.848, None, 86.932,
    'Taipei': [193.693, 310.595, None, 531.410, 467.136, 0.003, 556.076,
    'Sanzhong': [73.508, 141.726, 359.530, None, 190.253, None, 223.984,
    'Wugu': [99.106, 147.136, 419.402, 256.016, None, None, 151.079, 0.07
    'Freeway Bureau': [0.644, 2.198, 6.899, 2.516, 2.299, None, None, Non
    'Linkou North': [59.435, 84.535, 532.334, 218.264, 160.957, 899.149,
    'Linkou South': [0.016, 0.021, 0.128, 0.052, 0.054, 0.277, 0.209, Non
}

# Load traffic data into a DataFrame and set index, then convert traffic
traffic_df = pd.DataFrame(traffic_data)
traffic_df.set_index('From', inplace=True)
traffic_df = traffic_df / 60  # Convert to vehicles per minute

# Fill NaN values in journey times and traffic flows
for i in range(len(journey_df)):
    for j in range(len(journey_df.columns)):
        if i == j:  # Diagonal elements set to 0
            if pd.isna(journey_df.iloc[i, j]):
                journey_df.iloc[i, j] = 0
        else:  # Non-diagonal elements set to 100000
            if pd.isna(journey_df.iloc[i, j]):
                journey_df.iloc[i, j] = 1000

for i in range(len(traffic_df)):
    for j in range(len(traffic_df.columns)):
        if pd.isna(traffic_df.iloc[i, j]):
            traffic_df.iloc[i, j] = 0

# Calculate demand for each destination (sum of traffic flows to that are
demand = traffic_df.sum(axis=0)
total_demand = demand.sum()

# Display results
print("\n=== Journey-time matrix after NaN→0 or 100000 ===")
print(journey_df)

print("\n=== Traffic-flow matrix after NaN→0 ===")
print(traffic_df)

print("\n=== Demand per destination (veh/min) ===")
print(demand)
print(f"\nTotal demand (veh/min): {total_demand:.4f}")

# Optimization model using PuLP
def solve_hub_selection(journey_df, demand_series, hubs_required=2, minim
    regions = list(journey_df.index)
    problem = LpProblem("HubSelectionOptimization", LpMinimize)

    # Define binary decision variables
    hub_decision = {i: LpVariable(f"hub_{i}", 0, 1, LpBinary) for i in re
    service_decision = {(i, j): LpVariable(f"service_{i}_{j}", 0, 1, LpBi

    # (a) Hub selection constraint: We need exactly 'hubs_required' hubs
    problem += lpSum(hub_decision[i] for i in regions) == hubs_required

    # (b) Link service to hub decision: If area is not a hub, it can't se
```

```
    for i in regions:
        problem += lpSum(service_decision[i, j] for j in regions) <= len(

    # (c) Each area must be served by exactly one hub
    for j in regions:
        problem += lpSum(service_decision[i, j] for i in regions) == 1
        for i in regions:
            problem += service_decision[i, j] <= hub_decision[i]

    # (e) Load balance constraint: no hub serves more than 60% of the tot
    total_demand = demand_series.sum()
    for i in regions:
        problem += lpSum(demand_series[j] * service_decision[i, j] for j

    # Define objective function: minimize either total cost or maximum ti
    if minimize_max_time:  # (i) Minimax problem
        max_time = LpVariable("max_time", 0)
        for i in regions:
            for j in regions:
                problem += journey_df.loc[i, j] * service_decision[i, j]
        problem += max_time
    else:  # (h) Minimize total transportation cost
        problem += lpSum(journey_df.loc[i, j] * demand_series[j] * servic

    problem.solve(PULP_CBC_CMD(msg=False))

    selected_hubs = [i for i in regions if hub_decision[i].value() > 0.5]
    assignment = {j: next(i for i in regions if service_decision[i, j].va

    delivery_time = {j: journey_df.loc[assignment[j], j] for j in regions
    total_cost = sum(delivery_time[j] * demand_series[j] for j in regions
    average_time = total_cost / total_demand
    max_delivery_time = max(delivery_time.values())

    return selected_hubs, assignment, total_cost, average_time, max_deliv
```

```
=== Journey-time matrix after NaN→0 or 100000 ===
                  Neihu  Yuanshan  Taipei  Sanzhong     Wugu  Freeway Burea
u  \
From
Neihu              0.00      3.19   11.93     13.98    15.11            21.3
5
Yuanshan           7.39      0.00    8.26     10.18    11.91            18.0
8
Taipei             9.52      1.85    0.00      1.33     3.30             8.0
4
Sanzhong          12.69      4.93    3.13      0.00     2.21             8.0
2
Wugu              17.31      9.87    7.46      4.11     0.00             5.1
0
Freeway Bureau  169.33   1000.00  177.62   1000.00  1000.00             0.0
0
Linkou North      25.80     18.61   16.25     12.97     8.30          1000.0
0
Linkou South      45.43     31.94   26.90     22.29    12.64          1000.0
0

                Linkou North  Linkou South
From
Neihu                  22.24         30.01
Yuanshan               19.12         25.02
Taipei                  9.97         13.83
Sanzhong                9.24         14.12
Wugu                    6.82         11.63
Freeway Bureau          3.01          8.38
Linkou North            0.00          8.62
Linkou South            6.07          0.00

=== Traffic-flow matrix after NaN→0 ===
                   Neihu  Yuanshan    Taipei  Sanzhong      Wugu  \
From
Neihu           0.000000  2.194200  3.228217  1.225133  1.651767
Yuanshan        2.200267  0.000000  5.176583  2.362100  2.452267
Taipei          2.157017  4.793517  0.000000  5.992167  6.990033
Sanzhong        1.097150  2.845783  8.856833  0.000000  4.266933
Wugu            0.963317  2.047467  7.785600  3.170883  0.000000
Freeway Bureau  0.000083  0.000000  0.000050  0.000000  0.000000
Linkou North    0.630750  1.448867  9.267933  3.733067  2.517983
Linkou South    0.000133  0.000633  0.002483  0.001367  0.001183

                Freeway Bureau  Linkou North  Linkou South
From
Neihu                 0.010733      0.990583      0.000267
Yuanshan              0.036633      1.408917      0.000350
Taipei                0.114983      8.872233      0.002133
Sanzhong              0.041933      3.637733      0.000867
Wugu                  0.038317      2.682617      0.000900
Freeway Bureau        0.000000     14.985817      0.004617
Linkou North          0.000000      0.000000      0.003483
Linkou South          0.000000      0.003900      0.000000

=== Demand per destination (veh/min) ===
Neihu              7.048717
Yuanshan          13.330467
Taipei            34.317700
Sanzhong          16.484717
```

```
Wugu              17.880167
Freeway Bureau     0.242600
Linkou North      32.581800
Linkou South       0.012617
dtype: float64

Total demand (veh/min): 121.8988
```

In [2]:
```
# (h) Logistics – Minimize Total Cost
selected_hubs_logistics, assignment_logistics, cost_logistics, avg_time_l

print("===== (h) Logistics Plan – Minimize Total Cost =====")
print("Selected hubs:", selected_hubs_logistics)
print(f"Total cost          : {cost_logistics:.3f}")
print(f"Average time (min) : {avg_time_logistics:.3f}")
print(f"Max time (min)     : {max_time_logistics:.3f}")
```

```
===== (h) Logistics Plan – Minimize Total Cost =====
Selected hubs: ['Taipei', 'Linkou North']
Total cost         : 264.154
Average time (min) : 2.167
Max time (min)     : 9.520
```

In [3]:
```
# (i) Publicity – Minimize Max Time
selected_hubs_publicity, assignment_publicity, cost_publicity, avg_time_p

print("===== (i) Publicity Plan – Minimize Max Time =====")
print("Selected hubs:", selected_hubs_publicity)
print(f"Total cost          : {cost_publicity:.3f}")
print(f"Average time (min) : {avg_time_publicity:.3f}")
print(f"Max time (min)     : {max_time_publicity:.3f}")
```

```
===== (i) Publicity Plan – Minimize Max Time =====
Selected hubs: ['Taipei', 'Linkou North']
Total cost         : 264.154
Average time (min) : 2.167
Max time (min)     : 9.520
```

In [4]:
```
# (j) Current scenario with only one hub in Taipei
current_hub = "Taipei"
cost_current = sum(journey_df.loc[current_hub, j] * demand[j] for j in re
avg_time_current = cost_current / total_demand
max_time_current = max(journey_df.loc[current_hub, j] for j in regions)

# Improvement percentages
cost_improvement_pct = (cost_current - cost_logistics) / cost_current * 1
max_time_improvement_pct = (max_time_current - max_time_publicity) / max_

print("===== (j) Current Scenario – Taipei as Sole Hub =====")
print(f"Total cost          : {cost_current:.3f}")
print(f"Average time (min) : {avg_time_current:.3f}")
print(f"Max time (min)     : {max_time_current:.3f}")

print("\n--- Improvement over Current Scenario ---")
print(f"Cost ↓ by Logistics plan  : {cost_improvement_pct:.2f} %")
print(f"Max time ↓ by Publicity plan : {max_time_improvement_pct:.2f} %")
```

```
===== (j) Current Scenario – Taipei as Sole Hub =====
Total cost         : 499.660
Average time (min) : 4.099
Max time (min)     : 13.830

--- Improvement over Current Scenario ---
Cost ↓ by Logistics plan  : 47.13 %
Max time ↓ by Publicity plan : 31.16 %
```

In [5]:

```
# (k) Summary Table
summary_df = pd.DataFrame({
    "Scenario": ["Current", "Logistics (min cost)", "Publicity (min max-t
    "Hubs": [current_hub, selected_hubs_logistics, selected_hubs_publicit
    "Total cost": [cost_current, cost_logistics, cost_publicity],
    "Avg time (min)": [avg_time_current, avg_time_logistics, avg_time_pub
    "Max time (min)": [max_time_current, max_time_logistics, max_time_pub
})

print("\nSummary of the results:")
print(summary_df)
```

```
Summary of the results:
                  Scenario                     Hubs  Total cost  \
0                  Current                   Taipei  499.659908
1     Logistics (min cost)  [Taipei, Linkou North]  264.154462
2  Publicity (min max-time)  [Taipei, Linkou North]  264.154462

   Avg time (min)  Max time (min)
0        4.098974           13.83
1        2.166998            9.52
2        2.166998            9.52
```

(k) Discussion Based on the Results

Summary Table

Scenario	Hubs	Total cost	Avg time (min)	Max time (min)
Current	Taipei	499.66	4.10	13.83
Logistics (min cost)	Taipei, Linkou North	264.15	2.17	9.52
Publicity (min max-time)	Taipei, Linkou North	264.15	2.17	9.52

Improvement over the Current Scenario

- **Total transportation cost** decreased by **47.13%** under the new plan.
- **Maximum delivery time** decreased by **31.16%** under the new plan.

Discussion

Both the Logistics plan (minimizing total cost) and the Publicity plan (minimizing maximum delivery time) yield the same solution:

- **Selected hubs**: Taipei and Linkou North

- **Total transportation cost**: 264.15
- **Average delivery time**: 2.17 minutes
- **Maximum delivery time**: 9.52 minutes

Compared to the current single-hub setup (only Taipei as the hub), the two-hub solution offers significant improvements:

- Total cost reduction by 47.13%.
- Maximum delivery time reduction by 31.16%.
- Average delivery time reduction from 4.10 minutes to 2.17 minutes.

Conclusion

Since both optimization objectives lead to the same hub selection and performance metrics, there is no conflict between the two plans.
The company can simultaneously achieve both goals: **minimizing overall transportation cost** and **shortening the longest delivery time**.

Thus, implementing the new two-hub strategy with Taipei and Linkou North is strongly recommended, as it clearly improves both operational efficiency and service level compared to the current setup.