a) Optimization Problem Formulation

This optimization problem involves **assigning payloads to rockets** in such a way that the total cost is minimized while respecting several constraints, including rocket capacity, fuel consumption based on distances, and payload assignment.

Problem Description

We are tasked with launching a set of payloads to various space destinations using available rockets. Each rocket has a weight capacity, a limit on the fuel it can use, and payloads incur fuel costs based on the distance to the destination. The objective is to assign payloads to rockets to minimize the total mission cost while ensuring that all payloads are fully assigned and all constraints are respected.

Mathematical Formulation:

Sets

- N: Set of payloads, indexed by n, where $n = 1, 2, \ldots, N$.
- K: Set of rockets, indexed by k, where $k = 1, 2, \ldots, K$.
- M: Set of destinations, indexed by m, where $m = 1, 2, \ldots, M$.

Parameters

- W_n: Weight of payload n (constant for each payload).
- L_k: Maximum weight capacity of rocket k (the total weight a rocket can carry).
- C_k: Fixed cost of launching rocket k (constant for each rocket).
- R_{nk}: Fuel consumption for transporting payload n using rocket k, dependent on the distance to the assigned destination.
- FuelCap_k: Maximum fuel capacity of rocket k (total fuel available for a given rocket).
- D_m: Distance to destination m.

Decision Variables

- $x_{nk} \in \mathbb{R}^+$: **weight of payload n assigned to rocket k.**
- $y_k \in \mathbb{R}^+$: **total weight assigned to rocket k.**
- $z_m \in \mathbb{R}^+$: **total payload weight assigned to destination m.**

Objective Function

The objective is to minimize the total cost of the mission, which is composed of two parts:

1. **Rocket Launch Costs**: This is a fixed cost associated with launching each rocket. It is proportional to the total weight carried by the rocket.
2. **Fuel Consumption Costs**: The fuel consumption depends on the weight of payloads and the distance to the destination.

Thus, the objective function can be written as:

$$\min \sum_{k=1}^{K} C_k y_k + \sum_{n=1}^{N} \sum_{k=1}^{K} R_{nk} x_{nk}$$

Where:

- $C_k y_k$ is the cost of launching rocket k based on the total weight assigned to the rocket.
- $R_{nk} x_{nk}$ is the fuel consumption for transporting payload n using rocket k, which incorporates distance-based factors.

Constraints

1. **Payload Assignment Constraints**:

   Each payload must be fully assigned to rockets. The sum of the weights assigned to all rockets for each payload must equal the total weight of the payload:

$$\sum_{k=1}^{K} x_{nk} = W_n \quad \forall n \in N$$

   This ensures that each payload n is fully assigned to rockets.

1. **Rocket Capacity Constraints**:

   The total weight assigned to each rocket must not exceed its capacity:

$$\sum_{n=1}^{N} x_{nk} \leq L_k \quad \forall k \in K$$

   This ensures that no rocket is overloaded beyond its maximum capacity.

1. **Rocket Weight Consistency**:

The total weight assigned to a rocket k must match the sum of the weights of the payloads assigned to it:

$$y_k = \sum_{n=1}^{N} x_{nk} \quad \forall k \in K$$

1. **Fuel Capacity Constraints**:

   The total fuel consumed by the payloads assigned to each rocket must not exceed the rocket's fuel capacity:

$$\sum_{n=1}^{N} R_{nk} x_{nk} \leq \text{FuelCap}_k \quad \forall k \in K$$

   This ensures that the fuel requirements for each rocket are met and the rocket doesn't run out of fuel.

1. **Destination Assignment Constraints**:

   The total weight assigned to a destination must be greater than or equal to the sum of the payloads sent to that destination, scaled by the distance:

$$z_m \geq \sum_{n=1}^{N} \sum_{k=1}^{K} x_{nk} D_m \quad \forall m \in M$$

   This ensures that payload weights are accounted for according to the distance to each destination.

Thus, the optimization problem is:

$$\min \sum_{k=1}^{K} C_k y_k + \sum_{n=1}^{N} \sum_{k=1}^{K} R_{nk} x_{nk}$$

Subject to:

1. Payload Assignment:

$$\sum_{k=1}^{K} x_{nk} = W_n \quad \forall n \in N$$

1. Rocket Capacity:

$$\sum_{n=1}^{N} x_{nk} \leq L_k \quad \forall k \in K$$

1. Rocket Weight Consistency:

$$y_k = \sum_{n=1}^{N} x_{nk} \quad \forall k \in K$$

1. Fuel Capacity:

$$\sum_{n=1}^{N} R_{nk} x_{nk} \leq \text{FuelCap}_k \quad \forall k \in K$$

1. Destination Assignment:

$$z_m \geq \sum_{n=1}^{N} \sum_{k=1}^{K} x_{nk} D_m \quad \forall m \in M$$

In [1]:

```
from pyomo.environ import *
import random

random.seed(1234)


model = ConcreteModel()

# Parameters
N = 20  # Number of payloads
M = 4   # Number of destinations
K = 6   # Number of rockets

# Randomly generation of parameter values

C = [random.uniform(100, 200) for k in range(K)]  # Launch cost for each rocket

W = [random.uniform(5, 20) for n in range(N)]      # Weight of each payload

D = [random.uniform(1, 5) for m in range(M)]      # Distance to each destination

L = [random.uniform(50, 100) for k in range(K)]   # Capacity of each rocket

fuel_capacity = [random.uniform(200, 400) for k in range(K)]  # Fuel capacity for each rocket

# Fuel consumption per unit weight depends on the destination

# Each payload-rocket pair has a different fuel consumption based on the distance

R = [[random.uniform(0.5, 2.5) * D[random.randint(0, M - 1)] for k in range(K)] for n in range(N)]


model.x = Var(range(N), range(K), domain=NonNegativeReals)  # Weight of payload n assigned to rocket k
```

```python
model.y = Var(range(K), domain=NonNegativeReals)            # Total weight assigned to rocket k

model.z = Var(range(M), domain=NonNegativeReals)            # Total weight assigned to destination m

# Objective Function: Minimize total cost

def objective_rule(model):
    return sum(C[k] * model.y[k] for k in range(K)) + sum(R[n][k] * model.x[n, k]
           for n in range(N) for k in range(K))
model.obj = Objective(rule=objective_rule, sense=minimize)
```

In [2]:

```python
# Constraints

# Payload assignment constraint: the total weight of each payload must be fully assigned to rockets

def payload_assignment_rule(model, n):
    return sum(model.x[n, k] for k in range(K)) == W[n]
model.payload_assignment = Constraint(range(N), rule=payload_assignment_rule)

# Rocket capacity constraint: sum of weights assigned to a rocket cannot exceed its capacity

def rocket_capacity_rule(model, k):
    return sum(model.x[n, k] for n in range(N)) <= L[k]
model.rocket_capacity = Constraint(range(K), rule=rocket_capacity_rule)

# Rocket weight consistency: the total weight assigned to rocket k must match the sum of the payloads assigned to it

def rocket_weight_rule(model, k):
    return model.y[k] == sum(model.x[n, k] for n in range(N))
model.rocket_weight = Constraint(range(K), rule=rocket_weight_rule)

# Fuel requirement constraint: total fuel required by the payloads on each rocket must not exceed its fuel capacity

def fuel_requirement_rule(model, k):
    return sum(R[n][k] * model.x[n, k] for n in range(N)) <= fuel_capacity[k]
model.fuel_requirement = Constraint(range(K), rule=fuel_requirement_rule)

# Destination constraint: ensuring that payloads are assigned based on distances

def destination_selection_rule(model, m):
    return model.z[m] >= sum(model.x[n, k] * D[m] for n in range(N) for k in range(K))
model.destination_selection = Constraint(range(M), rule=destination_selection_rule)
```

In [3]:

```python
solver = SolverFactory('glpk')
result = solver.solve(model, tee=True)

model.display()
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmpqc039x9q.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmp9lg1u4_8.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmpe6b99bg5.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmpe6b99bg5.pyomo.lp'...
43 rows, 131 columns, 971 non-zeros
1364 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmp9lg1u4_8.glpk.glp'...
1311 lines were written
GLPK Simplex Optimizer 5.0
43 rows, 131 columns, 971 non-zeros
Preprocessing...
32 rows, 120 columns, 360 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  8.443e+00  ratio =  8.443e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 32
      0: obj =   3.561916065e+04 inf =   1.061e+03 (2)
     16: obj =   3.541047534e+04 inf =   0.000e+00 (0)
*    44: obj =   3.100609484e+04 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.2 Mb (217696 bytes)
Writing basic solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmpqc039x9q.glpk.raw'...
183 lines were written
Model unknown

  Variables:
    x : Size=120, Index=x_index
        Key     : Lower : Value            : Upper : Fixed : Stale : Domain
         (0, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (0, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (0, 2) :     0 : 3.72016848841337 :  None : False : False : NonNegativeReals
         (0, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (0, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (0, 5) :     0 : 11.3532837339064 :  None : False : False : NonNegativeReals
         (1, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (1, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (1, 2) :     0 : 6.25907340255624 :  None : False : False : NonNegativeReals
         (1, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (1, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (1, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (2, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (2, 1) :     0 : 16.4972139918769 :  None : False : False : NonNegativeReals
         (2, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (2, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (2, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (2, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (3, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (3, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (3, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (3, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (3, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (3, 5) :     0 : 8.55214663044677 :  None : False : False : NonNegativeReals
         (4, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (4, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (4, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (4, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (4, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (4, 5) :     0 : 5.46221032589916 :  None : False : False : NonNegativeReals
         (5, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (5, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (5, 2) :     0 : 16.8315907585443 :  None : False : False : NonNegativeReals
         (5, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (5, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (5, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (6, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (6, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (6, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (6, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (6, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (6, 5) :     0 : 10.1913344839568 :  None : False : False : NonNegativeReals
         (7, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (7, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (7, 2) :     0 : 14.3492221255874 :  None : False : False : NonNegativeReals
         (7, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (7, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (7, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (8, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (8, 1) :     0 : 14.2372354265543 :  None : False : False : NonNegativeReals
         (8, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (8, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (8, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (8, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (9, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (9, 1) :     0 : 7.22831958062431 :  None : False : False : NonNegativeReals
         (9, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (9, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (9, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (9, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (10, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (10, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (10, 2) :     0 : 7.74635971114897 :  None : False : False : NonNegativeReals
        (10, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (10, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (10, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (11, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (11, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (11, 2) :     0 : 2.53024219391521 :  None : False : False : NonNegativeReals
```

```
      (11, 3) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (11, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (11, 5) :     0 : 4.18595235141511 :  None : False : False : NonNegativeReals
      (12, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (12, 1) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (12, 2) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (12, 3) :     0 : 5.21928170730364 :  None : False : False : NonNegativeReals
      (12, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (12, 5) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (13, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (13, 1) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (13, 2) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (13, 3) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (13, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (13, 5) :     0 : 12.3012731090714 :  None : False : False : NonNegativeReals
      (14, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (14, 1) :     0 : 7.31403036737718 :  None : False : False : NonNegativeReals
      (14, 2) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (14, 3) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (14, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (14, 5) :     0 : 12.1594930463661 :  None : False : False : NonNegativeReals
      (15, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (15, 1) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (15, 2) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (15, 3) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (15, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (15, 5) :     0 : 5.96843421465779 :  None : False : False : NonNegativeReals
      (16, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (16, 1) :     0 : 2.09537339537195 :  None : False : False : NonNegativeReals
      (16, 2) :     0 :  11.020949387895 :  None : False : False : NonNegativeReals
      (16, 3) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (16, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (16, 5) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (17, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (17, 1) :     0 : 6.36783286692866 :  None : False : False : NonNegativeReals
      (17, 2) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (17, 3) :     0 :   5.620645518196 :  None : False : False : NonNegativeReals
      (17, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (17, 5) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (18, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (18, 1) :     0 : 14.0219517434158 :  None : False : False : NonNegativeReals
      (18, 2) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (18, 3) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (18, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (18, 5) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (19, 0) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (19, 1) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (19, 2) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (19, 3) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (19, 4) :     0 :               0.0 :  None : False : False : NonNegativeReals
      (19, 5) :     0 : 6.33393244985996 :  None : False : False : NonNegativeReals
  y : Size=6, Index=y_index
      Key : Lower : Value            : Upper : Fixed : Stale : Domain
        0 :     0 :              0.0 :  None : False : False : NonNegativeReals
        1 :     0 : 67.7619573721492 :  None : False : False : NonNegativeReals
        2 :     0 : 62.4576060680605 :  None : False : False : NonNegativeReals
        3 :     0 : 10.8399272254996 :  None : False : False : NonNegativeReals
        4 :     0 :              0.0 :  None : False : False : NonNegativeReals
        5 :     0 : 76.5080603455795 :  None : False : False : NonNegativeReals
  z : Size=4, Index=z_index
      Key : Lower : Value            : Upper : Fixed : Stale : Domain
        0 :     0 : 721.456336862261 :  None : False : False : NonNegativeReals
        1 :     0 : 452.179782427856 :  None : False : False : NonNegativeReals
        2 :     0 : 701.814229023914 :  None : False : False : NonNegativeReals
        3 :     0 : 778.573517524115 :  None : False : False : NonNegativeReals

Objectives:
  obj : Size=1, Index=None, Active=True
      Key  : Active : Value
      None :   True : 31006.094840149948

Constraints:
  payload_assignment : Size=20
      Key : Lower              : Body              : Upper
        0 : 15.073452222319776 : 15.07345222231977 : 15.073452222319776
        1 :  6.259073402556259 :  6.25907340255624 :  6.259073402556259
        2 : 16.497213991876944 :  16.4972139918769 : 16.497213991876944
        3 :  8.552146630446767 :  8.55214663044677 :  8.552146630446767
        4 :  5.462210325899149 :  5.46221032589916 :  5.462210325899149
        5 :  16.83159075854425 :  16.8315907585443 :  16.83159075854425
        6 : 10.191334483956847 :  10.1913344839568 : 10.191334483956847
        7 : 14.349222125587527 :  14.3492221255874 : 14.349222125587527
        8 : 14.237235426554228 :  14.2372354265543 : 14.237235426554228
        9 :  7.228319580624314 :  7.22831958062431 :  7.228319580624314
       10 :  7.746359711148974 :  7.74635971114897 :  7.746359711148974
       11 :  6.716194545330314 : 6.716194545330319 :  6.716194545330314
       12 :  5.219281707303637 :  5.21928170730364 :  5.219281707303637
       13 : 12.301273109071374 :  12.3012731090714 : 12.301273109071374
       14 : 19.473523413743237 : 19.47352341374328 : 19.473523413743237
       15 :  5.968434214657791 :  5.96843421465779 :  5.968434214657791
       16 : 13.116322783266954 : 13.11632278326695 : 13.116322783266954
       17 : 11.988478385124644 : 11.98847838512466 : 11.988478385124644
       18 : 14.021951743415771 :  14.0219517434158 : 14.021951743415771
       19 :  6.333932449859935 :  6.33393244985996 :  6.333932449859935
  rocket_capacity : Size=6
      Key : Lower : Body              : Upper
        0 :  None :               0.0 : 74.05181856832593
        1 :  None :  67.7619573721491 : 67.76195737214917
        2 :  None : 62.45760606806049 : 62.45760606806045
        3 :  None : 10.83992722549964 : 96.67577490211733
```

```
        4 :  None :               0.0 : 72.66940097382468
        5 :  None : 76.50806034557948 : 76.50806034557951
    rocket_weight : Size=6
        Key : Lower : Body                    : Upper
          0 :   0.0 :                     0.0 :   0.0
          1 :   0.0 :   9.947598300641403e-14 :   0.0
          2 :   0.0 :   7.105427357601002e-15 :   0.0
          3 :   0.0 :  -3.907985046680551e-14 :   0.0
          4 :   0.0 :                     0.0 :   0.0
          5 :   0.0 :  1.4210854715202004e-14 :   0.0
    fuel_requirement : Size=6
        Key : Lower : Body               : Upper
          0 :  None :                0.0 : 203.85991326194338
          1 :  None : 271.84536330494336 : 301.62038515595845
          2 :  None : 201.15604748354875 : 201.15604748354863
          3 :  None : 25.587008557165454 : 228.75368551911907
          4 :  None :                0.0 :  294.5653850694813
          5 :  None : 275.46948815451987 :  275.4694881545193
    destination_selection : Size=4
        Key : Lower : Body                    : Upper
          0 :  None :                     0.0 :   0.0
          1 :  None : -1.1368683772161603e-13 :   0.0
          2 :  None :   4.547473508864641e-13 :   0.0
          3 :  None : -1.1368683772161603e-13 :   0.0
```

In [4]:

```python
print("Optimal Payload Assignments (x[n,k]):")
for n in range(N):
    for k in range(K):
        print(f"x[{n},{k}] = {model.x[n,k].value}")

print("\nTotal Weight on Rockets (y[k]):")
for k in range(K):
    print(f"y[{k}] = {model.y[k].value}")

print("\nWeight Assigned to Destinations (z[m]):")
for m in range(M):
    print(f"z[{m}] = {model.z[m].value}")
```

```
Optimal Payload Assignments (x[n,k]):
x[0,0] = 0.0
x[0,1] = 0.0
x[0,2] = 3.72016848841337
x[0,3] = 0.0
x[0,4] = 0.0
x[0,5] = 11.3532837339064
x[1,0] = 0.0
x[1,1] = 0.0
x[1,2] = 6.25907340255624
x[1,3] = 0.0
x[1,4] = 0.0
x[1,5] = 0.0
x[2,0] = 0.0
x[2,1] = 16.4972139918769
x[2,2] = 0.0
x[2,3] = 0.0
x[2,4] = 0.0
x[2,5] = 0.0
x[3,0] = 0.0
x[3,1] = 0.0
x[3,2] = 0.0
x[3,3] = 0.0
x[3,4] = 0.0
x[3,5] = 8.55214663044677
x[4,0] = 0.0
x[4,1] = 0.0
x[4,2] = 0.0
x[4,3] = 0.0
x[4,4] = 0.0
x[4,5] = 5.46221032589916
x[5,0] = 0.0
x[5,1] = 0.0
x[5,2] = 16.8315907585443
x[5,3] = 0.0
x[5,4] = 0.0
x[5,5] = 0.0
x[6,0] = 0.0
x[6,1] = 0.0
x[6,2] = 0.0
x[6,3] = 0.0
x[6,4] = 0.0
x[6,5] = 10.1913344839568
x[7,0] = 0.0
x[7,1] = 0.0
x[7,2] = 14.3492221255874
x[7,3] = 0.0
x[7,4] = 0.0
x[7,5] = 0.0
x[8,0] = 0.0
x[8,1] = 14.2372354265543
x[8,2] = 0.0
x[8,3] = 0.0
x[8,4] = 0.0
x[8,5] = 0.0
x[9,0] = 0.0
x[9,1] = 7.22831958062431
x[9,2] = 0.0
x[9,3] = 0.0
x[9,4] = 0.0
x[9,5] = 0.0
x[10,0] = 0.0
x[10,1] = 0.0
x[10,2] = 7.74635971114897
x[10,3] = 0.0
x[10,4] = 0.0
x[10,5] = 0.0
x[11,0] = 0.0
x[11,1] = 0.0
x[11,2] = 2.53024219391521
x[11,3] = 0.0
x[11,4] = 0.0
x[11,5] = 4.18595235141511
x[12,0] = 0.0
x[12,1] = 0.0
x[12,2] = 0.0
x[12,3] = 5.21928170730364
x[12,4] = 0.0
x[12,5] = 0.0
x[13,0] = 0.0
x[13,1] = 0.0
x[13,2] = 0.0
x[13,3] = 0.0
x[13,4] = 0.0
x[13,5] = 12.3012731090714
x[14,0] = 0.0
x[14,1] = 7.31403036737718
x[14,2] = 0.0
x[14,3] = 0.0
x[14,4] = 0.0
x[14,5] = 12.1594930463661
x[15,0] = 0.0
x[15,1] = 0.0
x[15,2] = 0.0
x[15,3] = 0.0
x[15,4] = 0.0
x[15,5] = 5.96843421465779
x[16,0] = 0.0
x[16,1] = 2.09537339537195
x[16,2] = 11.020949387895
```

```
x[16,3] = 0.0
x[16,4] = 0.0
x[16,5] = 0.0
x[17,0] = 0.0
x[17,1] = 6.36783286692866
x[17,2] = 0.0
x[17,3] = 5.620645518196
x[17,4] = 0.0
x[17,5] = 0.0
x[18,0] = 0.0
x[18,1] = 14.0219517434158
x[18,2] = 0.0
x[18,3] = 0.0
x[18,4] = 0.0
x[18,5] = 0.0
x[19,0] = 0.0
x[19,1] = 0.0
x[19,2] = 0.0
x[19,3] = 0.0
x[19,4] = 0.0
x[19,5] = 6.33393244985996

Total Weight on Rockets (y[k]):
y[0] = 0.0
y[1] = 67.7619573721492
y[2] = 62.4576060680605
y[3] = 10.8399272254996
y[4] = 0.0
y[5] = 76.5080603455795

Weight Assigned to Destinations (z[m]):
z[0] = 721.456336862261
z[1] = 452.179782427856
z[2] = 701.814229023914
z[3] = 778.573517524115
```

b) Optimization Problem Interpretation

The solution obtained reveals the optimal payload assignments and the total cost associated with the rocket launches and payload transportation. The results confirm that the optimization model successfully minimizes the total cost while ensuring all payloads are assigned to rockets without violating any capacity, fuel, or destination constraints. The model effectively utilizes only rockets 1, 2, 3, and 5, suggesting that the payload distribution achieves efficiency by avoiding the use of rockets 0 and 4.

Summary of the Solution

- **Objective Value**: The total minimized cost of the mission is:

$$\text{Objective Value} = 31,006.0948$$

This value represents the total cost incurred for launching the rockets and transporting the payloads, including both launch and fuel costs.

Variables and Constraints

1. **Decision Variables**:

- x_{nk}: Weight of payload n assigned to rocket k. Some example values include:
  - $x[0,2] = 3.7202$
  - $x[0,5] = 11.3533$
  - $x[8,1] = 14.2372$
  - $x[13,5] = 12.3013$

  This shows how payloads are distributed across various rockets.

1. **Total Weight on Rockets**:
   - y_k: Total weight assigned to each rocket k.
     - $y[0] = 0.0$
     - $y[1] = 67.7620$
     - $y[2] = 62.4576$
     - $y[3] = 10.84$
     - $y[5] = 76.5081$

   Rockets 1, 2, 3, and 5 are used, while rockets 0 and 4 remain unused.

1. **Total Weight Assigned to Destinations**:

- z_m: Total weight assigned to destination m.
  - $z[0] = 721.4563$
  - $z[1] = 452.1798$
  - $z[2] = 701.8142$
  - $z[3] = 778.5735$

  These values represent the total weight delivered to each destination.

Constraints Verification

1. **Payload Assignment Constraints**:

Each payload is fully assigned to rockets. For instance:

$$\sum_{k=1}^{K} x[0,k] = W[0] = 15.0735$$

This ensures that the entire weight of each payload is allocated.

1. **Rocket Capacity Constraints**:

   Rockets are not overloaded:

$$\sum_{n=1}^{N} x[n,k] \leq L[k], \quad \forall k \in K$$

   Example:

   - $y[1] = 67.76196 \leq 67.7620$ (Capacity constraint is met).

1. **Rocket Weight Consistency**: The total weight assigned to each rocket matches the sum of payload weights:

$$y[k] = \sum_{n=1}^{N} x[n,k], \quad \forall k \in K$$

1. **Fuel Capacity Constraints**:

   Fuel requirements are within the limits:

$$\sum_{n=1}^{N} R_{nk} \cdot x[n,k] \leq \text{FuelCap}[k], \quad \forall k \in K$$

   Example:

   - $271.8454 \leq 301.6204$ for rocket 1.

1. **Destination Assignment Constraints**:

   The weight assigned meets the requirement for each destination:

$$z[m] \geq \sum_{n=1}^{N} \sum_{k=1}^{K} x[n,k] \cdot D[m], \quad \forall m \in M$$

   Example:

   - $z[0] = 721.4563$ satisfies the constraint.

c) Sensitivities

In [6]:

```
# Apply dual transformation to enable dual values
model.dual = Suffix(direction=Suffix.IMPORT)

# Solve the model using GLPK solver
solver = SolverFactory('glpk')
result = solver.solve(model, tee=True)
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmp94he_0xx.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmpbg1qledv.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmp71tawbmj.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmp71tawbmj.pyomo.lp'...
43 rows, 131 columns, 971 non-zeros
1364 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmpbg1qledv.glpk.glp'...
1311 lines were written
GLPK Simplex Optimizer 5.0
43 rows, 131 columns, 971 non-zeros
Preprocessing...
32 rows, 120 columns, 360 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  8.443e+00  ratio =  8.443e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 32
      0: obj =   3.561916065e+04 inf =   1.061e+03 (2)
     16: obj =   3.541047534e+04 inf =   0.000e+00 (0)
*    44: obj =   3.100609484e+04 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.2 Mb (217696 bytes)
Writing basic solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmp94he_0xx.glpk.raw'...
183 lines were written
```

In [7]:

```
model.display()
```

```
Model unknown

  Variables:
    x : Size=120, Index=x_index
        Key      : Lower : Value            : Upper : Fixed : Stale : Domain
          (0, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (0, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (0, 2) :     0 : 3.72016848841337 :  None : False : False : NonNegativeReals
          (0, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (0, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (0, 5) :     0 : 11.3532837339064 :  None : False : False : NonNegativeReals
          (1, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (1, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (1, 2) :     0 : 6.25907340255624 :  None : False : False : NonNegativeReals
          (1, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (1, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (1, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (2, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (2, 1) :     0 : 16.4972139918769 :  None : False : False : NonNegativeReals
          (2, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (2, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (2, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (2, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (3, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (3, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (3, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (3, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (3, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (3, 5) :     0 : 8.55214663044677 :  None : False : False : NonNegativeReals
          (4, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (4, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (4, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (4, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (4, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (4, 5) :     0 : 5.46221032589916 :  None : False : False : NonNegativeReals
          (5, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (5, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (5, 2) :     0 : 16.8315907585443 :  None : False : False : NonNegativeReals
          (5, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (5, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (5, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (6, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (6, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (6, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (6, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (6, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (6, 5) :     0 : 10.1913344839568 :  None : False : False : NonNegativeReals
          (7, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (7, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (7, 2) :     0 : 14.3492221255874 :  None : False : False : NonNegativeReals
          (7, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (7, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (7, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (8, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (8, 1) :     0 : 14.2372354265543 :  None : False : False : NonNegativeReals
          (8, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (8, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (8, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (8, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (9, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (9, 1) :     0 : 7.22831958062431 :  None : False : False : NonNegativeReals
          (9, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (9, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (9, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
          (9, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (10, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (10, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (10, 2) :     0 : 7.74635971114897 :  None : False : False : NonNegativeReals
         (10, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (10, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (10, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (11, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (11, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (11, 2) :     0 : 2.53024219391521 :  None : False : False : NonNegativeReals
         (11, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (11, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (11, 5) :     0 : 4.18595235141511 :  None : False : False : NonNegativeReals
         (12, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (12, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (12, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (12, 3) :     0 : 5.21928170730364 :  None : False : False : NonNegativeReals
         (12, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (12, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (13, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (13, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (13, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (13, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (13, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (13, 5) :     0 : 12.3012731090714 :  None : False : False : NonNegativeReals
         (14, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (14, 1) :     0 : 7.31403036737718 :  None : False : False : NonNegativeReals
         (14, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (14, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (14, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (14, 5) :     0 : 12.1594930463661 :  None : False : False : NonNegativeReals
         (15, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (15, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (15, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (15, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
         (15, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
```

```
        (15, 5) :     0 : 5.96843421465779 :  None : False : False : NonNegativeReals
        (16, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (16, 1) :     0 : 2.09537339537195 :  None : False : False : NonNegativeReals
        (16, 2) :     0 :  11.020949387895 :  None : False : False : NonNegativeReals
        (16, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (16, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (16, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (17, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (17, 1) :     0 : 6.36783286692866 :  None : False : False : NonNegativeReals
        (17, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (17, 3) :     0 :   5.620645518196 :  None : False : False : NonNegativeReals
        (17, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (17, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (18, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (18, 1) :     0 : 14.0219517434158 :  None : False : False : NonNegativeReals
        (18, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (18, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (18, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (18, 5) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (19, 0) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (19, 1) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (19, 2) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (19, 3) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (19, 4) :     0 :              0.0 :  None : False : False : NonNegativeReals
        (19, 5) :     0 : 6.33393244985996 :  None : False : False : NonNegativeReals
    y : Size=6, Index=y_index
        Key : Lower : Value            : Upper : Fixed : Stale : Domain
          0 :     0 :              0.0 :  None : False : False : NonNegativeReals
          1 :     0 : 67.7619573721492 :  None : False : False : NonNegativeReals
          2 :     0 : 62.4576060680605 :  None : False : False : NonNegativeReals
          3 :     0 : 10.8399272254996 :  None : False : False : NonNegativeReals
          4 :     0 :              0.0 :  None : False : False : NonNegativeReals
          5 :     0 : 76.5080603455795 :  None : False : False : NonNegativeReals
    z : Size=4, Index=z_index
        Key : Lower : Value            : Upper : Fixed : Stale : Domain
          0 :     0 : 721.456336862261 :  None : False : False : NonNegativeReals
          1 :     0 : 452.179782427856 :  None : False : False : NonNegativeReals
          2 :     0 : 701.814229023914 :  None : False : False : NonNegativeReals
          3 :     0 : 778.573517524115 :  None : False : False : NonNegativeReals

  Objectives:
    obj : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True : 31006.094840149948

  Constraints:
    payload_assignment : Size=20
        Key : Lower              : Body              : Upper
          0 : 15.073452222319776 : 15.07345222231977 : 15.073452222319776
          1 :  6.259073402556259 :  6.25907340255624 :  6.259073402556259
          2 : 16.497213991876944 :  16.4972139918769 : 16.497213991876944
          3 :  8.552146630446767 :  8.55214663044677 :  8.552146630446767
          4 :  5.462210325899149 :  5.46221032589916 :  5.462210325899149
          5 :  16.83159075854425 :  16.8315907585443 :  16.83159075854425
          6 : 10.191334483956847 :  10.1913344839568 : 10.191334483956847
          7 : 14.349222125587527 :  14.3492221255874 : 14.349222125587527
          8 : 14.237235426554228 :  14.2372354265543 : 14.237235426554228
          9 :  7.228319580624314 :  7.22831958062431 :  7.228319580624314
         10 :  7.746359711148974 :  7.74635971114897 :  7.746359711148974
         11 :  6.716194545330314 : 6.716194545330319 :  6.716194545330314
         12 :  5.219281707303637 :  5.21928170730364 :  5.219281707303637
         13 : 12.301273109071374 :  12.3012731090714 : 12.301273109071374
         14 : 19.473523413743237 : 19.47352341374328 : 19.473523413743237
         15 :  5.968434214657791 :  5.96843421465779 :  5.968434214657791
         16 : 13.116322783266954 : 13.11632278326695 : 13.116322783266954
         17 : 11.988478385124644 : 11.98847838512466 : 11.988478385124644
         18 : 14.021951743415771 :  14.0219517434158 : 14.021951743415771
         19 :  6.333932449859935 :  6.33393244985996 :  6.333932449859935
    rocket_capacity : Size=6
        Key : Lower : Body              : Upper
          0 :  None :               0.0 : 74.05181856832593
          1 :  None :  67.7619573721491 : 67.76195737214917
          2 :  None : 62.45760606806049 : 62.45760606806045
          3 :  None : 10.83992722549964 : 96.67577490211733
          4 :  None :               0.0 : 72.66940097382468
          5 :  None : 76.50806034557948 : 76.50806034557951
    rocket_weight : Size=6
        Key : Lower : Body                   : Upper
          0 :   0.0 :                    0.0 :   0.0
          1 :   0.0 :  9.947598300641403e-14 :   0.0
          2 :   0.0 :  7.105427357601002e-15 :   0.0
          3 :   0.0 : -3.907985046680551e-14 :   0.0
          4 :   0.0 :                    0.0 :   0.0
          5 :   0.0 : 1.4210854715202004e-14 :   0.0
    fuel_requirement : Size=6
        Key : Lower : Body               : Upper
          0 :  None :                0.0 : 203.85991326194338
          1 :  None : 271.84536330494336 : 301.62038515595845
          2 :  None : 201.15604748354875 : 201.15604748354863
          3 :  None : 25.587008557165454 : 228.75368551911907
          4 :  None :                0.0 :  294.5653850694813
          5 :  None : 275.46948815451987 :  275.4694881545193
    destination_selection : Size=4
        Key : Lower : Body                   : Upper
          0 :  None :                    0.0 :   0.0
          1 :  None : -1.1368683772161603e-13 :   0.0
          2 :  None :   4.547473508864641e-13 :   0.0
          3 :  None : -1.1368683772161603e-13 :   0.0
```

```
In [8]: # Display sensitivity analysis (dual values for constraints)
sensitivity_data = {
    'Payload Assignment': [model.dual[model.payload_assignment[n]] for n in range(N)],
    'Rocket Capacity': [model.dual[model.rocket_capacity[k]] for k in range(K)],
    'Rocket Weight Consistency': [model.dual[model.rocket_weight[k]] for k in range(K)],
    'Fuel Requirement': [model.dual[model.fuel_requirement[k]] for k in range(K)],
    'Destination Selection': [model.dual[model.destination_selection[m]] for m in range(M)]
}

print("Sensitivity Analysis:")
for constraint, sensitivities in sensitivity_data.items():
    print(f"\n{constraint} Sensitivities:")
    for index, value in enumerate(sensitivities):
        print(f"  Index {index}: {value}")
```

```
Sensitivity Analysis:

Payload Assignment Sensitivities:
  Index 0: 186.005428060539
  Index 1: 185.131072539536
  Index 2: 192.287084869319
  Index 3: 185.472680117716
  Index 4: 194.03787729468
  Index 5: 190.897524562869
  Index 6: 188.379798890956
  Index 7: 185.606181255534
  Index 8: 190.783814880163
  Index 9: 189.884960014366
  Index 10: 184.713599251672
  Index 11: 187.96478290446
  Index 12: 193.079472572346
  Index 13: 188.722633771207
  Index 14: 190.523611418006
  Index 15: 188.947607121921
  Index 16: 193.109124042265
  Index 17: 193.809569032719
  Index 18: 189.436541320442
  Index 19: 187.330574674952

Rocket Capacity Sensitivities:
  Index 0: 0.0
  Index 1: -43.0182551869882
  Index 2: -76.5743234224126
  Index 3: 0.0
  Index 4: 0.0
  Index 5: -25.1601455359739

Rocket Weight Consistency Sensitivities:
  Index 0: 196.645353569214
  Index 1: 144.073259917535
  Index 2: 100.749147005859
  Index 3: 191.097596244912
  Index 4: 193.926899736376
  Index 5: 158.222757305895

Fuel Requirement Sensitivities:
  Index 0: 0.0
  Index 1: 0.0
  Index 2: -2.41344281532693
  Index 3: 0.0
  Index 4: 0.0
  Index 5: -0.403714009186231

Destination Selection Sensitivities:
  Index 0: 0.0
  Index 1: 0.0
  Index 2: 0.0
  Index 3: 0.0
```

Sensitivity Analysis Interpretation

This sensitivity analysis reveals the dual values associated with each constraint in the optimization problem. These values indicate how much the objective function would change if the right-hand side of a constraint were relaxed or tightened by a small amount. Below is a detailed interpretation of the results.

Payload Assignment Sensitivities

The dual values for the **Payload Assignment** constraints represent the marginal contribution of each payload's weight requirement to the overall objective function. Here are the values:

$$\begin{aligned} \text{Index 0:} &\quad 186.0054 \\ \text{Index 1:} &\quad 185.1311 \\ \text{Index 2:} &\quad 192.2871 \\ \text{Index 3:} &\quad 185.4727 \\ \text{Index 4:} &\quad 194.0379 \\ &\vdots \\ \text{Index 19:} &\quad 187.3306 \end{aligned}$$

These values reflect the importance of each payload in terms of cost impact. For example, **Index 4** has one of the highest values (194.0379), meaning that increasing the weight requirement for this payload would significantly increase the objective. High dual values in this category indicate payloads that are costly to accommodate due to high fuel or weight requirements, or due to their alignment with limited rocket capacities.

Rocket Capacity Sensitivities

For **Rocket Capacity** constraints, the dual values show the marginal cost impact of adding one unit of capacity to each rocket:

$$\begin{aligned}
&\text{Index 0:} \quad 0.0 \\
&\text{Index 1:} \quad -43.0183 \\
&\text{Index 2:} \quad -76.5743 \\
&\text{Index 3:} \quad 0.0 \\
&\text{Index 4:} \quad 0.0 \\
&\text{Index 5:} \quad -25.1601
\end{aligned}$$

Rockets **Index 1, 2, and 5** have negative dual values, indicating that increasing the capacity of these rockets would reduce the total cost. Rocket **Index 2** has the largest impact, with a dual value of -76.5743, suggesting that a capacity increase for this rocket is particularly beneficial. This insight can guide operational decisions, showing that shifting payloads to certain rockets or enhancing their capacity would lead to cost savings.

Rocket Weight Consistency Sensitivities

The **Rocket Weight Consistency** constraint dual values indicate how the total payload weight assigned to each rocket affects costs:

$$\begin{aligned}
&\text{Index 0:} \quad 196.6454 \\
&\text{Index 1:} \quad 144.0733 \\
&\text{Index 2:} \quad 100.7491 \\
&\text{Index 3:} \quad 191.0976 \\
&\text{Index 4:} \quad 193.9269 \\
&\text{Index 5:} \quad 158.2228
\end{aligned}$$

These values reveal that **Rockets 0, 3, and 4** have particularly high sensitivities, with values close to or above 190. This suggests that these rockets are near their capacity limits, and any additional payload weight assigned to them would substantially increase the objective cost. Increasing the capacity or assigning fewer payloads to these rockets could alleviate these cost pressures.

Fuel Requirement Sensitivities

The **Fuel Requirement** constraint sensitivities indicate how marginal changes in fuel capacity for each rocket would impact the cost:

$$\begin{aligned}
&\text{Index 0:} \quad 0.0 \\
&\text{Index 1:} \quad 0.0 \\
&\text{Index 2:} \quad -2.4134 \\
&\text{Index 3:} \quad 0.0 \\
&\text{Index 4:} \quad 0.0 \\
&\text{Index 5:} \quad -0.4037
\end{aligned}$$

Only **Indexes 2 and 5** show non-zero dual values, at -2.4134 and -0.4037 respectively. These small negative values suggest that relaxing the fuel constraints for these rockets would provide a slight reduction in total costs, though the impact is minimal compared to other constraints.

Destination Selection Sensitivities

For the **Destination Selection** constraints, all dual values are zero:

$$\begin{aligned}
&\text{Index 0:} \quad 0.0 \\
&\text{Index 1:} \quad 0.0 \\
&\text{Index 2:} \quad 0.0 \\
&\text{Index 3:} \quad 0.0
\end{aligned}$$

These values indicate that the destination selection constraints are not currently impacting the cost objective. It suggests that the payload-to-destination assignments do not restrict the solution, likely due to ample capacity across destinations.

Summary of Important Findings

- **High Sensitivity Payloads**: Payloads indexed at **4 and 16** have high marginal costs, suggesting a focus on either reducing their weight or optimizing their rocket allocation.
- **Key Rockets for Cost Efficiency**: Rockets **Index 1, 2, and 5** show significant potential for cost savings if capacity or fuel requirements are adjusted, particularly **Rocket 2** for capacity.
- **Rocket Weight Constraints**: High sensitivity for **Rockets 0, 3, and 4** in weight consistency indicates that these rockets are close to optimal capacity and may need prioritization in payload assignments.

d) Dual Problem

1. **Introduce Dual Variables:**
   - For each constraint, we introduce a dual variable:
     - Let λ_n be the dual variable associated with the **payload assignment constraint**.
     - Let μ_k be the dual variable for the **rocket capacity constraint**.
     - Let ν_k be the dual variable for the **rocket weight consistency constraint**.
     - Let σ_k be the dual variable for the **fuel capacity constraint**.
     - Let θ_m be the dual variable for the **destination assignment constraint**.
2. **Dual Objective Function:** Since the primal is a minimization problem, the dual will be a maximization problem. The objective function of the dual problem is constructed by summing the products of each primal constraint's right-hand side with its corresponding dual variable:

$$\max \sum_{n=1}^{N} \lambda_n W_n + \sum_{k=1}^{K} \mu_k L_k + \sum_{k=1}^{K} \sigma_k \text{FuelCap}_k + \sum_{m=1}^{M} \theta_m \sum_{n=1}^{N} \sum_{k=1}^{K} x_{nk} D_m$$

3. **Dual Constraints:** The dual constraints correspond to each primal variable:

   - **Dual Constraints for x_{nk}:** For each payload assignment x_{nk}, we require:

$$\lambda_n + \mu_k + \nu_k + \sigma_k R_{nk} \geq C$$

In [43]:

```
dual_model = ConcreteModel()

N = 20  # Number of payloads
K = 6   # Number of rockets
M = 4   # Number of destinations


C = [random.uniform(100, 200) for k in range(K)]  # Launch cost for each rocket

W = [random.uniform(5, 20) for n in range(N)]      # Weight of each payload

D = [random.uniform(1, 5) for m in range(M)]      # Distance to each destination

L = [random.uniform(50, 100) for k in range(K)]   # Capacity of each rocket

fuel_capacity = [random.uniform(200, 400) for k in range(K)]  # Fuel capacity for each rocket

# Fuel consumption per unit weight depends on the destination

# Each payload-rocket pair has a different fuel consumption based on the distance

R = [[random.uniform(0.5, 2.5) * D[random.randint(0, M - 1)] for k in range(K)] for n in range(N)]

# Dual variables
dual_model.lambda_ = Var(range(N), domain=NonNegativeReals)  # Dual variable for payload assignment constraint
dual_model.mu = Var(range(K), domain=NonNegativeReals)       # Dual variable for rocket capacity constraint
dual_model.nu = Var(range(K), domain=Reals)                  # Dual variable for rocket weight consistency
dual_model.sigma = Var(range(K), domain=NonNegativeReals)    # Dual variable for fuel capacity constraint
dual_model.theta = Var(range(M), domain=NonNegativeReals)    # Dual variable for destination assignment constraint

# Objective: Maximize the dual objective
dual_model.obj = Objective(
    expr=sum(dual_model.lambda_[n] * W[n] for n in range(N)) +
         sum(dual_model.mu[k] * L[k] for k in range(K)) +
         sum(dual_model.sigma[k] * fuel_capacity[k] for k in range(K)) +
         sum(dual_model.theta[m] * sum(D[m] * R[n][k] for n in range(N) for k in range(K)) for m in range(M)),
    sense=maximize
)

# Dual constraints for each primal variable
dual_model.dual_constraints = ConstraintList()

# Constraints associated with x[n, k]
for n in range(N):
    for k in range(K):
        dual_model.dual_constraints.add(
            dual_model.lambda_[n] + dual_model.mu[k] + dual_model.nu[k] + dual_model.sigma[k] * R[n][k] >= C[k]
        )

# Constraints associated with y[k] (dual consistency for total weight on rockets)
for k in range(K):
    dual_model.dual_constraints.add(
        dual_model.nu[k] >= 0
    )

# Constraints associated with z[m] (dual consistency for destination weight requirements)
for m in range(M):
    dual_model.dual_constraints.add(
        dual_model.theta[m] >= 0
    )

# Solve the dual model using GLPK solver
solver = SolverFactory('glpk')
results = solver.solve(dual_model, tee=True)

# Display results
dual_model.display()
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmpms3bfmj6.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmp1x95z17c.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmp005qd515.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmp005qd515.pyomo.lp'...
131 rows, 43 columns, 491 non-zeros
970 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmp1x95z17c.glpk.glp'...
841 lines were written
GLPK Simplex Optimizer 5.0
131 rows, 43 columns, 491 non-zeros
Preprocessing...
PROBLEM HAS NO DUAL FEASIBLE SOLUTION
If you need actual output for non-optimal solution, use --nopresol
Time used:   0.0 secs
Memory used: 0.1 Mb (118388 bytes)
Writing basic solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmpms3bfmj6.glpk.raw'...
183 lines were written
Model unknown

  Variables:
    lambda_ : Size=20, Index=lambda__index
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :  None :  None : False :  True : NonNegativeReals
          1 :     0 :  None :  None : False :  True : NonNegativeReals
          2 :     0 :  None :  None : False :  True : NonNegativeReals
          3 :     0 :  None :  None : False :  True : NonNegativeReals
          4 :     0 :  None :  None : False :  True : NonNegativeReals
          5 :     0 :  None :  None : False :  True : NonNegativeReals
          6 :     0 :  None :  None : False :  True : NonNegativeReals
          7 :     0 :  None :  None : False :  True : NonNegativeReals
          8 :     0 :  None :  None : False :  True : NonNegativeReals
          9 :     0 :  None :  None : False :  True : NonNegativeReals
         10 :     0 :  None :  None : False :  True : NonNegativeReals
         11 :     0 :  None :  None : False :  True : NonNegativeReals
         12 :     0 :  None :  None : False :  True : NonNegativeReals
         13 :     0 :  None :  None : False :  True : NonNegativeReals
         14 :     0 :  None :  None : False :  True : NonNegativeReals
         15 :     0 :  None :  None : False :  True : NonNegativeReals
         16 :     0 :  None :  None : False :  True : NonNegativeReals
         17 :     0 :  None :  None : False :  True : NonNegativeReals
         18 :     0 :  None :  None : False :  True : NonNegativeReals
         19 :     0 :  None :  None : False :  True : NonNegativeReals
    mu : Size=6, Index=mu_index
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :  None :  None : False :  True : NonNegativeReals
          1 :     0 :  None :  None : False :  True : NonNegativeReals
          2 :     0 :  None :  None : False :  True : NonNegativeReals
          3 :     0 :  None :  None : False :  True : NonNegativeReals
          4 :     0 :  None :  None : False :  True : NonNegativeReals
          5 :     0 :  None :  None : False :  True : NonNegativeReals
    nu : Size=6, Index=nu_index
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :  None :  None :  None : False :  True :  Reals
          1 :  None :  None :  None : False :  True :  Reals
          2 :  None :  None :  None : False :  True :  Reals
          3 :  None :  None :  None : False :  True :  Reals
          4 :  None :  None :  None : False :  True :  Reals
          5 :  None :  None :  None : False :  True :  Reals
    sigma : Size=6, Index=sigma_index
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :  None :  None : False :  True : NonNegativeReals
          1 :     0 :  None :  None : False :  True : NonNegativeReals
          2 :     0 :  None :  None : False :  True : NonNegativeReals
          3 :     0 :  None :  None : False :  True : NonNegativeReals
          4 :     0 :  None :  None : False :  True : NonNegativeReals
          5 :     0 :  None :  None : False :  True : NonNegativeReals
    theta : Size=4, Index=theta_index
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :  None :  None : False :  True : NonNegativeReals
          1 :     0 :  None :  None : False :  True : NonNegativeReals
          2 :     0 :  None :  None : False :  True : NonNegativeReals
          3 :     0 :  None :  None : False :  True : NonNegativeReals

  Objectives:
    obj : Size=1, Index=None, Active=True
ERROR: evaluating object as numeric value: lambda_[0]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object lambda_[0]
ERROR: evaluating object as numeric value: obj
        (object: <class 'pyomo.core.base.objective.ScalarObjective'>)
    No value for uninitialized NumericValue object lambda_[0]
        Key : Active : Value
        None :   None :  None

  Constraints:
    dual_constraints : Size=130
        Key : Lower              : Body : Upper
          1 : 106.51927360600492 : None :  None
          2 :  114.0017657639388 : None :  None
          3 :  190.5360168929006 : None :  None
          4 : 158.80174661560315 : None :  None
          5 : 198.01687624222535 : None :  None
          6 : 175.35802629064867 : None :  None
          7 : 106.51927360600492 : None :  None
          8 :  114.0017657639388 : None :  None
          9 :  190.5360168929006 : None :  None
         10 : 158.80174661560315 : None :  None
         11 : 198.01687624222535 : None :  None
         12 : 175.35802629064867 : None :  None
```

```
 13 : 106.51927360600492 : None :  None
 14 :  114.0017657639388 : None :  None
 15 :  190.5360168929006 : None :  None
 16 : 158.80174661560315 : None :  None
 17 : 198.01687624222535 : None :  None
 18 : 175.35802629064867 : None :  None
 19 : 106.51927360600492 : None :  None
 20 :  114.0017657639388 : None :  None
 21 :  190.5360168929006 : None :  None
 22 : 158.80174661560315 : None :  None
 23 : 198.01687624222535 : None :  None
 24 : 175.35802629064867 : None :  None
 25 : 106.51927360600492 : None :  None
 26 :  114.0017657639388 : None :  None
 27 :  190.5360168929006 : None :  None
 28 : 158.80174661560315 : None :  None
 29 : 198.01687624222535 : None :  None
 30 : 175.35802629064867 : None :  None
 31 : 106.51927360600492 : None :  None
 32 :  114.0017657639388 : None :  None
 33 :  190.5360168929006 : None :  None
 34 : 158.80174661560315 : None :  None
 35 : 198.01687624222535 : None :  None
 36 : 175.35802629064867 : None :  None
 37 : 106.51927360600492 : None :  None
 38 :  114.0017657639388 : None :  None
 39 :  190.5360168929006 : None :  None
 40 : 158.80174661560315 : None :  None
 41 : 198.01687624222535 : None :  None
 42 : 175.35802629064867 : None :  None
 43 : 106.51927360600492 : None :  None
 44 :  114.0017657639388 : None :  None
 45 :  190.5360168929006 : None :  None
 46 : 158.80174661560315 : None :  None
 47 : 198.01687624222535 : None :  None
 48 : 175.35802629064867 : None :  None
 49 : 106.51927360600492 : None :  None
 50 :  114.0017657639388 : None :  None
 51 :  190.5360168929006 : None :  None
 52 : 158.80174661560315 : None :  None
 53 : 198.01687624222535 : None :  None
 54 : 175.35802629064867 : None :  None
 55 : 106.51927360600492 : None :  None
 56 :  114.0017657639388 : None :  None
 57 :  190.5360168929006 : None :  None
 58 : 158.80174661560315 : None :  None
 59 : 198.01687624222535 : None :  None
 60 : 175.35802629064867 : None :  None
 61 : 106.51927360600492 : None :  None
 62 :  114.0017657639388 : None :  None
 63 :  190.5360168929006 : None :  None
 64 : 158.80174661560315 : None :  None
 65 : 198.01687624222535 : None :  None
 66 : 175.35802629064867 : None :  None
 67 : 106.51927360600492 : None :  None
 68 :  114.0017657639388 : None :  None
 69 :  190.5360168929006 : None :  None
 70 : 158.80174661560315 : None :  None
 71 : 198.01687624222535 : None :  None
 72 : 175.35802629064867 : None :  None
 73 : 106.51927360600492 : None :  None
 74 :  114.0017657639388 : None :  None
 75 :  190.5360168929006 : None :  None
 76 : 158.80174661560315 : None :  None
 77 : 198.01687624222535 : None :  None
 78 : 175.35802629064867 : None :  None
 79 : 106.51927360600492 : None :  None
 80 :  114.0017657639388 : None :  None
 81 :  190.5360168929006 : None :  None
 82 : 158.80174661560315 : None :  None
 83 : 198.01687624222535 : None :  None
 84 : 175.35802629064867 : None :  None
 85 : 106.51927360600492 : None :  None
 86 :  114.0017657639388 : None :  None
 87 :  190.5360168929006 : None :  None
 88 : 158.80174661560315 : None :  None
 89 : 198.01687624222535 : None :  None
 90 : 175.35802629064867 : None :  None
 91 : 106.51927360600492 : None :  None
 92 :  114.0017657639388 : None :  None
 93 :  190.5360168929006 : None :  None
 94 : 158.80174661560315 : None :  None
 95 : 198.01687624222535 : None :  None
 96 : 175.35802629064867 : None :  None
 97 : 106.51927360600492 : None :  None
 98 :  114.0017657639388 : None :  None
 99 :  190.5360168929006 : None :  None
100 : 158.80174661560315 : None :  None
101 : 198.01687624222535 : None :  None
102 : 175.35802629064867 : None :  None
103 : 106.51927360600492 : None :  None
104 :  114.0017657639388 : None :  None
105 :  190.5360168929006 : None :  None
106 : 158.80174661560315 : None :  None
107 : 198.01687624222535 : None :  None
108 : 175.35802629064867 : None :  None
109 : 106.51927360600492 : None :  None
110 :  114.0017657639388 : None :  None
111 :  190.5360168929006 : None :  None
112 : 158.80174661560315 : None :  None
```

```
113 : 198.01687624222535 : None :  None
114 : 175.35802629064867 : None :  None
115 : 106.51927360600492 : None :  None
116 :  114.0017657639388 : None :  None
117 :  190.5360168929006 : None :  None
118 : 158.80174661560315 : None :  None
119 : 198.01687624222535 : None :  None
120 : 175.35802629064867 : None :  None
121 :                0.0 : None :  None
122 :                0.0 : None :  None
123 :                0.0 : None :  None
124 :                0.0 : None :  None
125 :                0.0 : None :  None
126 :                0.0 : None :  None
127 :                0.0 : None :  None
128 :                0.0 : None :  None
129 :                0.0 : None :  None
130 :                0.0 : None :  None
```

The obtained solution is infeasible. I guess I haven't formulated the dual problem correctly and the constraints might be to restrictive, but I can't see how should it be done then. I hope this isn't a very degrading issue.

e) Integer Optimization Problem

This optimization problem builds upon the first one, with several modifications that introduce additional constraints and new decision variables. This reformulated model introduces binary variables, priority levels, and enhanced destination requirements.

Problem Formulation Summary

Sets

The sets remain consistent with the original problem:

- N: Set of payloads, indexed by $n = 1, 2, \ldots, N$.
- K: Set of rockets, indexed by $k = 1, 2, \ldots, K$.
- M: Set of destinations, indexed by $m = 1, 2, \ldots, M$.

Parameters

While the parameters in the second problem have similar roles as in the first problem, some key values are now **fixed constants** for the sake of model stability:

- **Payload Weight** (W_n): Each payload has a fixed weight of 10, denoted by $W_n = 10 \, \forall n \in N$.
- **Rocket Capacity** (L_k): Each rocket k has a distinct, predefined weight capacity, with values [50, 60, 70, 80, 90, 100].
- **Fuel Capacity** (FuelCap_k): Each rocket has a defined fuel capacity, with values [300, 350, 250, 300, 400, 450].
- **Launch Costs** (C_k): Each rocket has a fixed launch cost, with values [500, 600, 700, 800, 900, 1000].
- **Distances** (D_m): Distances to each destination are given as [1, 2, 3, 4].
- **Fuel Requirements** (R_{nk}): Fuel consumption per payload-rocket pair is defined as a function of payload and rocket indices, using $R_{nk} = 1 + 0.1 \times (n + k)$, which allows variability in fuel requirements.

Decision Variables

The second model introduces two additional decision variables:

1. **Binary Rocket Activation Variable** (b_k): b_k is a binary variable indicating whether rocket k is activated $(b_k = 1)$ or not $(b_k = 0)$. This variable enables more complex activation constraints, ensuring rockets are only activated if they meet minimum payload requirements.
2. **Priority Levels for Destinations** (p_m): p_m is an integer variable that denotes the priority level for each destination m. This allows payloads to be assigned to destinations according to priority requirements.

The original decision variables are still present:

- **Payload Assignment Weight** (x_{nk}): Weight of payload n assigned to rocket k.
- **Rocket Weight** (y_k): Total weight assigned to rocket k.
- **Destination Weight** (z_m): Total weight assigned to destination m.

Objective Function

The objective function remains similar, minimizing the total cost of the mission, which consists of:

1. **Rocket Launch Costs**: A fixed cost associated with launching each rocket, scaled by the total weight carried by the rocket.
2. **Fuel Consumption Costs**: Fuel costs based on payload weights and distances to destinations.

The objective function can be expressed as:

$$\min \sum_{k=1}^{K} C_k \cdot y_k + \sum_{n=1}^{N} \sum_{k=1}^{K} R_{nk} \cdot x_{nk}$$

Where:

- $C_k \cdot y_k$ represents the launch cost for rocket k.
- $R_{nk} \cdot x_{nk}$ represents the fuel consumption for transporting payload n using rocket k.

Constraints

The second model introduces several new constraints in addition to the ones from the first problem. Here is a summary of all constraints, highlighting the new or modified ones.

1. **Payload Assignment Constraint** (unchanged):

Each payload must be fully assigned to rockets. The total weight assigned to rockets for each payload n must equal W_n:

$$\sum_{k=1}^{K} x_{nk} = W_n \quad \forall n \in N$$

2. **Rocket Capacity Constraint** (unchanged):

The total weight assigned to each rocket must not exceed its capacity L_k:

$$\sum_{n=1}^{N} x_{nk} \leq L_k \quad \forall k \in K$$

3. **Rocket Weight Consistency** (unchanged):

The total weight assigned to rocket k must match the sum of the weights of the payloads assigned to it:

$$y_k = \sum_{n=1}^{N} x_{nk} \quad \forall k \in K$$

4. **Fuel Capacity Constraint** (unchanged):

The total fuel consumed by the payloads assigned to each rocket must not exceed its fuel capacity FuelCap_k:

$$\sum_{n=1}^{N} R_{nk} x_{nk} \leq \text{FuelCap}_k \quad \forall k \in K$$

5. **Rocket Activation Constraint** (new):

To enforce that rockets are only activated if they carry a minimum weight, a **large constant** M_{large} is introduced to link payload assignment with activation status:

$$\sum_{n=1}^{N} x_{nk} \leq M_{\text{large}} \cdot b_k \quad \forall k \in K$$

This constraint ensures that $b_k = 1$ only if the rocket is used, as M_{large} provides an upper bound for the maximum possible payload weight on the rocket.

6. **Minimum Payload Requirement for Activated Rockets** (new):

If a rocket is activated $(b_k = 1)$, it must carry at least a minimum weight W_{min}:

$$y_k \geq W_{\text{min}} \cdot b_k \quad \forall k \in K$$

This constraint helps avoid underutilization of rockets by ensuring that each activated rocket carries a significant payload.

7. **Priority Assignment Constraint for Destinations** (new):

Each destination m has an assigned priority level, and the total weight assigned to each destination must meet a minimum requirement proportional to the priority:

$$z_m \geq P_{\text{min}} \cdot p_m \quad \forall m \in M$$

Additionally, the total weight assigned to each destination is calculated based on payload assignments:

$$z_m = \sum_{n=1}^{N} \sum_{k=1}^{K} x_{nk} \cdot D_m \quad \forall m \in M$$

This constraint ensures that destinations with higher priority levels receive a correspondingly higher total payload weight.

In [15]:
```
model = ConcreteModel()

# Sets
N = 20  # Number of payloads
M = 4   # Number of destinations
K = 6   # Number of rockets

# Parameters
weights = [10] * N  # All payloads have weight 10
rocket_capacity = [50, 60, 70, 80, 90, 100]  # Capacity for each rocket
fuel_capacity = [300, 350, 250, 300, 400, 450]  # Fuel capacities for rockets
launch_costs = [500, 600, 700, 800, 900, 1000]  # Launch cost per rocket
distances = [1, 2, 3, 4]  # Distances to each destination
fuel_requirements = [[1 + 0.1 * (i + j) for j in range(K)] for i in range(N)]  # fuel rates

# Decision Variables
model.x = Var(range(N), range(K), domain=NonNegativeReals)  # Payload assignments (weight)
model.y = Var(range(K), domain=NonNegativeReals)            # Total weight on each rocket
model.z = Var(range(M), domain=NonNegativeReals)            # Total weight to each destination
model.b = Var(range(K), domain=Binary)                      # Binary rocket activation
model.p = Var(range(M), domain=NonNegativeIntegers)         # Priority levels for each destination

# Objective Function: Minimize total cost
model.obj = Objective(
```

```
    expr=sum(launch_costs[k] * model.y[k] for k in range(K)) +
         sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N) for k in range(K)),
    sense=minimize
)

# Constraints

# Payload assignment - each payload must be fully assigned to rockets
model.payload_assignment = ConstraintList()
for n in range(N):
    model.payload_assignment.add(sum(model.x[n, k] for k in range(K)) == weights[n])

# Rocket capacity constraints
model.rocket_capacity = ConstraintList()
for k in range(K):
    model.rocket_capacity.add(sum(model.x[n, k] for n in range(N)) <= rocket_capacity[k])

# Fuel constraints - fuel used per rocket must not exceed fuel capacity
model.fuel_capacity = ConstraintList()
for k in range(K):
    model.fuel_capacity.add(sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N)) <= fuel_capacity[k])

# Rocket weight consistency - total weight on rocket equals sum of payload weights assigned
model.rocket_weight_consistency = ConstraintList()
for k in range(K):
    model.rocket_weight_consistency.add(model.y[k] == sum(model.x[n, k] for n in range(N)))

# Rocket activation constraint (using a large constant for activation)
M_large = 100
model.rocket_activation = ConstraintList()
for k in range(K):
    model.rocket_activation.add(sum(model.x[n, k] for n in range(N)) <= M_large * model.b[k])

# Minimum payload requirement for activated rockets
W_min = 15
model.min_payload_requirement = ConstraintList()
for k in range(K):
    model.min_payload_requirement.add(model.y[k] >= W_min * model.b[k])

# Priority assignment constraint for destinations
# Ensuring each destination weight meets its priority level (if priority is non-zero)
P_min = 30  # Minimum payload per priority level
model.priority_assignment = ConstraintList()
for m in range(M):
    # Define destination weight based on total payload assigned to it across rockets
    model.priority_assignment.add(model.z[m] >= P_min * model.p[m])
    model.priority_assignment.add(model.z[m] == sum(model.x[n, k] for n in range(N) for k in range(K) if distances[m]))

# Solve the model using GLPK solver
solver = SolverFactory('glpk')
results = solver.solve(model, tee=True)

# Display results to check adjustments
optimal_payload_assignments = {(n, k): model.x[n, k].value for n in range(N) for k in range(K)}
optimal_rocket_weights = {k: model.y[k].value for k in range(K)}
optimal_destination_weights = {m: model.z[m].value for m in range(M)}
optimal_cost = model.obj()

print("Optimal Payload Assignments:", optimal_payload_assignments)
print("Total Weight on Rockets:", optimal_rocket_weights)
print("Weight Assigned to Destinations:", optimal_destination_weights)
print("Optimal Cost:", optimal_cost)
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmpc2m4oqbu.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmpptsrmr2g.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmplxcvsxyw.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmplxcvsxyw.pyomo.lp'...
C:\Users\Gabriel\AppData\Local\Temp\tmplxcvsxyw.pyomo.lp:1574: warning: lower bound of variable 'x131' redefined
C:\Users\Gabriel\AppData\Local\Temp\tmplxcvsxyw.pyomo.lp:1574: warning: upper bound of variable 'x131' redefined
59 rows, 141 columns, 1117 non-zeros
10 integer variables, 6 of which are binary
1580 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmpptsrmr2g.glpk.glp'...
1630 lines were written
GLPK Integer Optimizer 5.0
59 rows, 141 columns, 1117 non-zeros
10 integer variables, 6 of which are binary
Preprocessing...
58 rows, 140 columns, 1116 non-zeros
10 integer variables, 6 of which are binary
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  1.000e+02  ratio =  1.000e+02
GM: min|aij| =  6.585e-01  max|aij| =  1.519e+00  ratio =  2.306e+00
EQ: min|aij| =  4.526e-01  max|aij| =  1.000e+00  ratio =  2.210e+00
2N: min|aij| =  4.688e-01  max|aij| =  1.563e+00  ratio =  3.333e+00
Constructing initial basis...
Size of triangular part is 58
Solving LP relaxation...
GLPK Simplex Optimizer 5.0
58 rows, 140 columns, 1116 non-zeros
      0: obj =   1.944840000e+05 inf =   2.800e+02 (3)
     22: obj =   1.574470000e+05 inf =   0.000e+00 (0)
*    47: obj =   1.264160000e+05 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Integer optimization begins...
Long-step dual simplex will be used
+    47: mip =     not found yet >=              -inf        (1; 0)
+    51: >>>>>   1.264160000e+05 >=   1.264160000e+05   0.0% (5; 0)
+    51: mip =   1.264160000e+05 >=     tree is empty   0.0% (0; 9)
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.3 Mb (334410 bytes)
Writing MIP solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmpc2m4oqbu.glpk.raw'...
209 lines were written
Optimal Payload Assignments: {(0, 0): 10.0, (0, 1): 0.0, (0, 2): 0.0, (0, 3): 0.0, (0, 4): 0.0, (0, 5): 0.0, (1, 0): 0.0, (1, 1): 0.0, (1,
2): 10.0, (1, 3): 0.0, (1, 4): 0.0, (1, 5): 0.0, (2, 0): 0.0, (2, 1): 10.0, (2, 2): 0.0, (2, 3): 0.0, (2, 4): 0.0, (2, 5): 0.0, (3, 0): 10.
0, (3, 1): 0.0, (3, 2): 0.0, (3, 3): 0.0, (3, 4): 0.0, (3, 5): 0.0, (4, 0): 10.0, (4, 1): 0.0, (4, 2): 0.0, (4, 3): 0.0, (4, 4): 0.0, (4,
5): 0.0, (5, 0): 10.0, (5, 1): 0.0, (5, 2): 0.0, (5, 3): 0.0, (5, 4): 0.0, (5, 5): 0.0, (6, 0): 10.0, (6, 1): 0.0, (6, 2): 0.0, (6, 3): 0.
0, (6, 4): 0.0, (6, 5): 0.0, (7, 0): 0.0, (7, 1): 10.0, (7, 2): 0.0, (7, 3): 0.0, (7, 4): 0.0, (7, 5): 0.0, (8, 0): 0.0, (8, 1): 10.0, (8,
2): 0.0, (8, 3): 0.0, (8, 4): 0.0, (8, 5): 0.0, (9, 0): 0.0, (9, 1): 10.0, (9, 2): 0.0, (9, 3): 0.0, (9, 4): 0.0, (9, 5): 0.0, (10, 0): 0.
0, (10, 1): 10.0, (10, 2): 0.0, (10, 3): 0.0, (10, 4): 0.0, (10, 5): 0.0, (11, 0): 0.0, (11, 1): 0.0, (11, 2): 10.0, (11, 3): 0.0, (11, 4):
0.0, (11, 5): 0.0, (12, 0): 0.0, (12, 1): 10.0, (12, 2): 0.0, (12, 3): 0.0, (12, 4): 0.0, (12, 5): 0.0, (13, 0): 0.0, (13, 1): 0.0, (13,
2): 10.0, (13, 3): 0.0, (13, 4): 0.0, (13, 5): 0.0, (14, 0): 0.0, (14, 1): 0.0, (14, 2): 0.0, (14, 3): 10.0, (14, 4): 0.0, (14, 5): 0.0, (1
5, 0): 0.0, (15, 1): 0.0, (15, 2): 0.0, (15, 3): 10.0, (15, 4): 0.0, (15, 5): 0.0, (16, 0): 0.0, (16, 1): 0.0, (16, 2): 10.0, (16, 3): 0.0,
(16, 4): 0.0, (16, 5): 0.0, (17, 0): 0.0, (17, 1): 0.0, (17, 2): 10.0, (17, 3): 0.0, (17, 4): 0.0, (17, 5): 0.0, (18, 0): 0.0, (18, 1): 0.
0, (18, 2): 10.0, (18, 3): 0.0, (18, 4): 0.0, (18, 5): 0.0, (19, 0): 0.0, (19, 1): 0.0, (19, 2): 10.0, (19, 3): 0.0, (19, 4): 0.0, (19, 5):
0.0}
Total Weight on Rockets: {0: 50.0, 1: 60.0, 2: 70.0, 3: 20.0, 4: 0.0, 5: 0.0}
Weight Assigned to Destinations: {0: 200.0, 1: 200.0, 2: 200.0, 3: 200.0}
Optimal Cost: 126416.0
```

The following analysis breaks down the optimal assignments and evaluates the constraints.

1. Payload Assignments

Each payload has been assigned fully to a rocket, with exactly 10 units allocated per payload as specified by the model parameters. The assignment pattern shows that payloads are distributed among rockets in a way that satisfies the total capacity constraint. Each payload $x_{n,k}$ is assigned to only one rocket, as intended.

The payload assignment constraint for each payload n is given by:

$$\sum_{k=1}^{K} x_{n,k} = \text{weights}[n] \quad \forall n \in \{1, \dots, N\}$$

where:

- $x_{n,k}$ represents the weight of payload n assigned to rocket k.
- $\text{weights}[n] = 10$ for each payload.

2. Total Weight on Rockets

The total weight on each rocket y_k is calculated based on the assigned payloads and is shown below:

- Rockets 0, 1, 2, and 3 have total weights of 50.0, 60.0, 70.0, and 20.0 respectively, matching the assigned payloads.
- Rockets 4 and 5 have a total weight of 0.0, indicating they are unused in this solution.

This satisfies the rocket capacity constraint, which ensures that the weight assigned to each rocket does not exceed its capacity:

$$\sum_{n=1}^{N} x_{n,k} \le L[k] \quad \forall k \in \{1, \dots, K\}$$

where $\text{L}[k]$ represents the capacity for rocket k.

Additionally, the total weight consistency constraint ensures that the total weight y_k assigned to each rocket is the sum of the payload weights assigned to it:

$$y_k = \sum_{n=1}^{N} x_{n,k} \quad \forall k \in \{1, \ldots, K\}$$

The **rocket activation constraint** limits the model to use only necessary rockets:

$$\sum_{n=1}^{N} x_{n,k} \leq M_{\text{large}} \cdot b_k \quad \forall k \in \{1, \ldots, K\}$$

where:

- M_{large} is a large constant for ensuring b_k becomes 1 only if the rocket k is used.
- b_k is a binary variable that indicates whether rocket k is active ($b_k = 1$) or not ($b_k = 0$).

The **minimum payload requirement** constraint ensures that if a rocket is active, it carries a minimum weight W_{min}:

$$y_k \geq W_{\text{min}} \cdot b_k \quad \forall k \in \{1, \ldots, K\}$$

where $W_{\text{min}} = 15$ is the minimum required payload.

3. Weight Assigned to Destinations

Each destination z_m has a total weight of 200.0, indicating that the payloads meet the **priority assignment constraint**. This ensures that each destination receives the required payload weight based on its priority level.

The constraint for the priority assignment to destinations is:

$$z_m \geq P_{\text{min}} \cdot p_m \quad \forall m \in \{1, \ldots, M\}$$

where:

- $P_{\text{min}} = 30$ is the minimum payload weight per priority level.
- p_m represents the priority level assigned to destination m.

The destination weight z_m is calculated as the sum of payload weights assigned to each destination:

$$z_m = \sum_{n=1}^{N} \sum_{k=1}^{K} x_{n,k} \cdot d_{n,m}$$

where $d_{n,m}$ is a parameter indicating if payload n is assigned to destination m.

4. Optimal Cost

The optimal cost Z is calculated by minimizing the total launch and fuel costs:

$$Z = \sum_{k=1}^{K} \text{C\_k}[k] \cdot y_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \text{R}[n][k] \cdot x_{n,k}$$

where:

- $C_k[k]$ is the fixed launch cost for rocket k.
- $\text{R}[n][k]$ represents the fuel requirement per unit weight for payload n on rocket k.

The resulting optimal cost is $126,416.0$, which is consistent with the objective function, payload assignments, and capacity constraints, indicating that the cost minimization is effective.

f) Relaxed Problem

To define the relaxed version of the previous problem, we will remove the integer and binary constraints. This means we will allow all variables, including the originally binary variables b_k and integer variables p_m to take any real values within their bounds.

In [16]:

```
model = ConcreteModel()

# Sets
N = 20  # Number of payloads
M = 4   # Number of destinations
K = 6   # Number of rockets

# Parameters
weights = [10] * N  # All payloads have weight 10
rocket_capacity = [50, 60, 70, 80, 90, 100]  # Capacity for each rocket
fuel_capacity = [300, 350, 250, 300, 400, 450]  # Fuel capacities for rockets
launch_costs = [500, 600, 700, 800, 900, 1000]  # Launch cost per rocket
distances = [1, 2, 3, 4]  # Distances to each destination
fuel_requirements = [[1 + 0.1 * (i + j) for j in range(K)] for i in range(N)]  # fuel rates

# Decision Variables
model.x = Var(range(N), range(K), domain=NonNegativeReals)  # Payload assignments (weight)
model.y = Var(range(K), domain=NonNegativeReals)            # Total weight on each rocket
model.z = Var(range(M), domain=NonNegativeReals)            # Total weight to each destination

# Relaxed variables (originally binary or integer)
model.b = Var(range(K), domain=NonNegativeReals, bounds=(0, 1))  # Relaxed binary rocket activation
model.p = Var(range(M), domain=NonNegativeReals)                 # Relaxed integer priority levels for destinations
```

```python
# Objective Function: Minimize total cost
model.obj = Objective(
    expr=sum(launch_costs[k] * model.y[k] for k in range(K)) +
         sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N) for k in range(K)),
    sense=minimize
)

# Constraints

# Payload assignment - each payload must be fully assigned to rockets
model.payload_assignment = ConstraintList()
for n in range(N):
    model.payload_assignment.add(sum(model.x[n, k] for k in range(K)) == weights[n])

# Rocket capacity constraints
model.rocket_capacity = ConstraintList()
for k in range(K):
    model.rocket_capacity.add(sum(model.x[n, k] for n in range(N)) <= rocket_capacity[k])

# Fuel constraints - fuel used per rocket must not exceed fuel capacity
model.fuel_capacity = ConstraintList()
for k in range(K):
    model.fuel_capacity.add(sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N)) <= fuel_capacity[k])

# Rocket weight consistency - total weight on rocket equals sum of payload weights assigned
model.rocket_weight_consistency = ConstraintList()
for k in range(K):
    model.rocket_weight_consistency.add(model.y[k] == sum(model.x[n, k] for n in range(N)))

# Rocket activation constraint (relaxed)
M_large = 100
model.rocket_activation = ConstraintList()
for k in range(K):
    model.rocket_activation.add(sum(model.x[n, k] for n in range(N)) <= M_large * model.b[k])

# Minimum payload requirement for activated rockets (relaxed)
W_min = 15
model.min_payload_requirement = ConstraintList()
for k in range(K):
    model.min_payload_requirement.add(model.y[k] >= W_min * model.b[k])

# Priority assignment constraint for destinations (relaxed)
P_min = 30
model.priority_assignment = ConstraintList()
for m in range(M):
    model.priority_assignment.add(model.z[m] >= P_min * model.p[m])
    model.priority_assignment.add(model.z[m] == sum(model.x[n, k] for n in range(N) for k in range(K) if distances[m]))

# Solve the relaxed model using a solver like GLPK
solver = SolverFactory('glpk')
results = solver.solve(model, tee=True)
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmpc3irpl7g.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmpoc078bz5.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmp6tpy4k2r.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmp6tpy4k2r.pyomo.lp'...
59 rows, 141 columns, 1117 non-zeros
1568 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmpoc078bz5.glpk.glp'...
1501 lines were written
GLPK Simplex Optimizer 5.0
59 rows, 141 columns, 1117 non-zeros
Preprocessing...
58 rows, 140 columns, 1116 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  1.000e+02  ratio =  1.000e+02
GM: min|aij| =  6.585e-01  max|aij| =  1.519e+00  ratio =  2.306e+00
EQ: min|aij| =  4.526e-01  max|aij| =  1.000e+00  ratio =  2.210e+00
Constructing initial basis...
Size of triangular part is 58
      0: obj =   1.944840000e+05 inf =   2.589e+02 (3)
     15: obj =   1.604500000e+05 inf =   1.865e-14 (0)
*    29: obj =   1.264160000e+05 inf =   9.177e-14 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.3 Mb (290168 bytes)
Writing basic solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmpc3irpl7g.glpk.raw'...
209 lines were written
Optimal Payload Assignments (Relaxed): {(0, 0): -2.72480477446915e-14, (0, 1): 1.82001720235518e-14, (0, 2): 0.0, (0, 3): 9.99999999999999,
(0, 4): 0.0, (0, 5): 0.0, (1, 0): 0.0, (1, 1): 0.0, (1, 2): 10.0, (1, 3): 0.0, (1, 4): 0.0, (1, 5): 0.0, (2, 0): 10.0, (2, 1): 0.0, (2, 2):
0.0, (2, 3): 0.0, (2, 4): 0.0, (2, 5): 0.0, (3, 0): 10.0, (3, 1): 0.0, (3, 2): 0.0, (3, 3): 0.0, (3, 4): 0.0, (3, 5): 0.0, (4, 0): 10.0,
(4, 1): 0.0, (4, 2): 0.0, (4, 3): 0.0, (4, 4): 0.0, (4, 5): 0.0, (5, 0): 10.0, (5, 1): 0.0, (5, 2): 0.0, (5, 3): 0.0, (5, 4): 0.0, (5, 5):
0.0, (6, 0): 10.0, (6, 1): 0.0, (6, 2): -7.41543550958815e-15, (6, 3): 0.0, (6, 4): 0.0, (6, 5): 0.0, (7, 0): 0.0, (7, 1): 0.0, (7, 2): 10.
0, (7, 3): 0.0, (7, 4): 0.0, (7, 5): 0.0, (8, 0): 0.0, (8, 1): 0.0, (8, 2): 10.0, (8, 3): 0.0, (8, 4): 0.0, (8, 5): 0.0, (9, 0): 0.0, (9,
1): 0.0, (9, 2): 10.0, (9, 3): 0.0, (9, 4): 0.0, (9, 5): 0.0, (10, 0): 0.0, (10, 1): 0.0, (10, 2): 10.0, (10, 3): 0.0, (10, 4): 0.0, (10,
5): 0.0, (11, 0): 0.0, (11, 1): 0.0, (11, 2): 10.0, (11, 3): 0.0, (11, 4): 0.0, (11, 5): 0.0, (12, 0): 0.0, (12, 1): 10.0, (12, 2): 0.0, (1
2, 3): 0.0, (12, 4): 0.0, (12, 5): 0.0, (13, 0): 0.0, (13, 1): 10.0, (13, 2): 0.0, (13, 3): 0.0, (13, 4): 0.0, (13, 5): 0.0, (14, 0): 0.0,
(14, 1): 10.0, (14, 2): 0.0, (14, 3): 0.0, (14, 4): 0.0, (14, 5): 0.0, (15, 0): 0.0, (15, 1): 10.0, (15, 2): 0.0, (15, 3): 0.0, (15, 4): 0.
0, (15, 5): 0.0, (16, 0): 0.0, (16, 1): 0.0, (16, 2): 10.0, (16, 3): 0.0, (16, 4): 0.0, (16, 5): 0.0, (17, 0): 0.0, (17, 1): 10.0, (17, 2):
0.0, (17, 3): 0.0, (17, 4): 0.0, (17, 5): 0.0, (18, 0): 0.0, (18, 1): 9.99999999999996, (18, 2): 0.0, (18, 3): 0.0, (18, 4): 0.0, (18, 5):
0.0, (19, 0): 0.0, (19, 1): 0.0, (19, 2): 0.0, (19, 3): 10.0, (19, 4): 0.0, (19, 5): -1.23023864064206e-14}
Total Weight on Rockets (Relaxed): {0: 50.0, 1: 60.0, 2: 70.0, 3: 20.0, 4: 0.0, 5: -1.23023864064206e-14}
Weight Assigned to Destinations (Relaxed): {0: 200.0, 1: 200.0, 2: 200.0, 3: 200.0}
Optimal Cost (Relaxed): 126415.99999999999
```

In [ ]:

```python
# Display relaxed results
optimal_payload_assignments = {(n, k): model.x[n, k].value for n in range(N) for k in range(K)}
optimal_rocket_weights = {k: model.y[k].value for k in range(K)}
optimal_destination_weights = {m: model.z[m].value for m in range(M)}
optimal_cost = model.obj()
```

```
print("Optimal Payload Assignments (Relaxed):", optimal_payload_assignments)
print("Total Weight on Rockets (Relaxed):", optimal_rocket_weights)
print("Weight Assigned to Destinations (Relaxed):", optimal_destination_weights)
print("Optimal Cost (Relaxed):", optimal_cost)
```

1. Optimal Payload Assignments

In the relaxed solution, each payload assignment is close to an integer value, specifically around 10 units for each assigned payload. Unlike the integer-constrained solution, there are small non-integer values due to numerical precision. For example:

- Payload assignment values include values like 9.99999999999999, $1.82001720235518 \times 10^{-14}$, and $-2.72480477446915 \times 10^{-14}$.
- These values are effectively zero or integer values but appear as non-integers due to the relaxation.

The payload assignment constraint ensures each payload is fully allocated across rockets, as described by:

$$\sum_{k=1}^{K} x_{n,k} = \text{weights}[n] \quad \forall n \in \{1, \ldots, N\}$$

where:

- $x_{n,k}$ represents the weight of payload n assigned to rocket k.
- $\text{weights}[n] = 10$ for each payload.

The small deviations from integer values do not affect feasibility but highlight the impact of removing integrality constraints on variable precision.

2. Total Weight on Rockets

The total weight on each rocket y_k in the relaxed solution is nearly identical to the integer-constrained solution:

- Rockets 0, 1, 2, and 3 carry weights of 50.0, 60.0, 70.0, and 20.0 units, respectively, matching the integer-constrained solution.
- Rockets 4 and 5 have weights close to zero, specifically 0.0 and $-1.23023864064206 \times 10^{-14}$, indicating they are effectively unused in this solution as well.

The total weight constraint ensures the sum of payload weights on each rocket does not exceed its capacity:

$$\sum_{n=1}^{N} x_{n,k} \leq \text{rocket\_capacity}[k] \quad \forall k \in \{1, \ldots, K\}$$

where $\text{rocket\_capacity}[k]$ is the capacity for rocket k.

The total weight consistency constraint, which requires y_k to equal the sum of payload weights assigned to rocket k, also holds true:

$$y_k = \sum_{n=1}^{N} x_{n,k} \quad \forall k \in \{1, \ldots, K\}$$

In summary, the relaxed solution maintains rocket weight consistency, but the weights include minor numerical deviations from exact values due to the lack of integer restrictions.

3. Weight Assigned to Destinations (Relaxed)

The relaxed solution assigns exactly 200.0 units to each destination z_m, identical to the integer-constrained solution. This outcome demonstrates that the **priority assignment constraints** are met effectively in both solutions, ensuring that each destination receives the required payload based on its priority level.

The priority assignment constraint is given by:

$$z_m \geq P_{\min} \cdot p_m \quad \forall m \in \{1, \ldots, M\}$$

where:

- $P_{\min} = 30$ is the minimum payload per priority level.
- p_m represents the priority level assigned to destination m.

The weight to each destination z_m is calculated based on the sum of payload weights directed to it:

$$z_m = \sum_{n=1}^{N} \sum_{k=1}^{K} x_{n,k} \cdot d_{n,m}$$

where $d_{n,m}$ is a parameter indicating if payload n is assigned to destination m.

The relaxed solution's destination weights are thus identical to the integer-constrained solution, showing that the relaxation does not impact destination weight assignments.

4. Optimal Cost

The optimal cost Z in the relaxed solution is:

$$Z = 126,415.99999999999$$

which is effectively the same as the integer-constrained solution:

$$Z = 126,416.0$$

This minor difference is due to numerical precision and rounding, with the relaxed solution approximating the integer-constrained solution. The objective function minimizes total launch and fuel costs:

$$Z = \sum_{k=1}^{K} \text{launch\_costs}[k] \cdot y_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \text{fuel\_requirements}[n][k] \cdot x_{n,k}$$

where:

- $\text{launch\\_costs}[k]$ is the fixed launch cost per rocket k.
- $\text{fuel\\_requirements}[n][k]$ represents the fuel requirement per unit weight for payload n on rocket k.

Since the optimal cost is nearly identical in both solutions, this suggests that the integer constraints did not significantly alter the feasible region for this problem.

Comparison Summary

Aspect	Integer-Constrained Solution	Relaxed Solution
Payload Assignments	Exact integer values	Close to integers with minor deviations
Rocket Weights	Exact values	Close to exact, with very minor deviations
Destination Weights	Exact values (200.0)	Exact values (200.0)
Optimal Cost	$126,416.0$	$126,415.99999999999$

1. **Precision Differences**:
    - The relaxed solution includes very small non-integer values close to zero in payload assignments and rocket weights. These deviations arise from the absence of integer constraints but do not affect the feasibility or interpretation of the solution.
2. **Cost Comparison**:
    - The optimal cost is nearly identical in both cases, indicating that the integer constraints did not significantly impact the objective. This often occurs when the feasible region's shape is not strongly altered by the integrality conditions.
3. **Constraint Satisfaction**:
    - Both solutions satisfy all constraints, including payload assignments, rocket capacities, destination weights, and minimum payload requirements for activated rockets. This confirms that the relaxation maintains the feasibility of the solution.

So, the relaxed solution provides a nearly identical outcome to the integer-constrained solution, suggesting that the integrality constraints have a limited impact on the overall solution structure in this case. The linear relaxation serves as a good approximation of the original problem.

g) Solving Multiple Instances

In [35]:
```python
from pyomo.environ import *
import random

# Function to generate and solve an instance of the integer-constrained problem
def solve_instance():

    # Generate random values for the problem size
    num_payloads = random.randint(10, 50)       # Random number of payloads between 10 and 50
    num_rockets = random.randint(3, 15)         # Random number of rockets between 3 and 15
    num_destinations = random.randint(2, 10)    # Random number of destinations between 2 and 10

    # Initialize the Pyomo model
    model = ConcreteModel()

    # Sets
    N = num_payloads       # Number of payloads
    K = num_rockets        # Number of rockets
    M = num_destinations   # Number of destinations


    # Seed selection only for the parameters so the comparison between instances makes sense
    random.seed(1234)
    # Parameters
    weights = [random.randint(5, 15) for _ in range(N)]
    rocket_capacity = [random.randint(50, 100) for _ in range(K)]
    fuel_capacity = [random.randint(200, 400) for _ in range(K)]
    launch_costs = [random.randint(400, 1000) for _ in range(K)]
    distances = [random.randint(1, 5) for _ in range(M)]
    fuel_requirements = [[random.uniform(0.5, 2.5) * distances[random.randint(0, M-1)] for _ in range(K)] for _ in range(N)]

    # Decision Variables
    model.x = Var(range(N), range(K), domain=NonNegativeIntegers)  # Weight of payload n assigned to rocket k
    model.y = Var(range(K), domain=NonNegativeIntegers)            # Total weight on each rocket
    model.z = Var(range(M), domain=NonNegativeIntegers)            # Total weight to each destination
    model.b = Var(range(K), domain=Binary)                         # Binary variable for rocket activation
    model.p = Var(range(M), domain=NonNegativeIntegers)            # Priority levels for destinations

    # Objective Function: Minimize total cost
    model.obj = Objective(
        expr=sum(launch_costs[k] * model.y[k] for k in range(K)) +
             sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N) for k in range(K)),
        sense=minimize
    )

    # Constraints
```

```python
    # Payload assignment - each payload must be fully assigned to rockets
    model.payload_assignment = ConstraintList()
    for n in range(N):
        model.payload_assignment.add(sum(model.x[n, k] for k in range(K)) == weights[n])

    # Rocket capacity constraints
    model.rocket_capacity = ConstraintList()
    for k in range(K):
        model.rocket_capacity.add(sum(model.x[n, k] for n in range(N)) <= rocket_capacity[k])

    # Fuel constraints - fuel used per rocket must not exceed fuel capacity
    model.fuel_capacity = ConstraintList()
    for k in range(K):
        model.fuel_capacity.add(sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N)) <= fuel_capacity[k])

    # Rocket weight consistency - total weight on rocket equals sum of payload weights assigned
    model.rocket_weight_consistency = ConstraintList()
    for k in range(K):
        model.rocket_weight_consistency.add(model.y[k] == sum(model.x[n, k] for n in range(N)))

    # Rocket activation constraint (using a large constant for activation)
    M_large = 100
    model.rocket_activation = ConstraintList()
    for k in range(K):
        model.rocket_activation.add(sum(model.x[n, k] for n in range(N)) <= M_large * model.b[k])

    # Minimum payload requirement for activated rockets
    W_min = 15
    model.min_payload_requirement = ConstraintList()
    for k in range(K):
        model.min_payload_requirement.add(model.y[k] >= W_min * model.b[k])

    # Priority assignment constraint for destinations
    P_min = 30
    model.priority_assignment = ConstraintList()
    for m in range(M):
        model.priority_assignment.add(model.z[m] >= P_min * model.p[m])
        model.priority_assignment.add(model.z[m] == sum(model.x[n, k] for n in range(N) for k in range(K) if distances[m]))

    # Solve the model using GLPK solver
    solver = SolverFactory('glpk')
    results = solver.solve(model, tee=True)

    # Check if a feasible solution was found
    # This avoids an error output if the solution is not feasible
    if (results.solver.termination_condition == TerminationCondition.optimal or
        results.solver.termination_condition == TerminationCondition.feasible):

        # Display results
        optimal_payload_assignments = {(n, k): model.x[n, k].value for n in range(N) for k in range(K)}
        optimal_rocket_weights = {k: model.y[k].value for k in range(K)}
        optimal_destination_weights = {m: model.z[m].value for m in range(M)}
        optimal_cost = model.obj()

        print(f"Instance with {N} payloads, {K} rockets, and {M} destinations:")
        print("Optimal Payload Assignments:", optimal_payload_assignments)
        print("Total Weight on Rockets:", optimal_rocket_weights)
        print("Weight Assigned to Destinations:", optimal_destination_weights)
        print("Optimal Cost:", optimal_cost)
        print("-" * 40)

    else:
        print(f"Instance with {N} payloads, {K} rockets, and {M} destinations is infeasible.")
        print("-" * 40)
```

In [37]:
```python
random.seed(22)
solve_instance()
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmpdh24sp6z.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmpx8c7ake_.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmp5hfco4st.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmp5hfco4st.pyomo.lp'...
C:\Users\Gabriel\AppData\Local\Temp\tmp5hfco4st.pyomo.lp:1312: warning: lower bound of variable 'x117' redefined
C:\Users\Gabriel\AppData\Local\Temp\tmp5hfco4st.pyomo.lp:1312: warning: upper bound of variable 'x117' redefined
53 rows, 125 columns, 787 non-zeros
124 integer variables, 6 of which are binary
1318 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmpx8c7ake_.glpk.glp'...
1246 lines were written
GLPK Integer Optimizer 5.0
53 rows, 125 columns, 787 non-zeros
124 integer variables, 6 of which are binary
Preprocessing...
52 rows, 124 columns, 786 non-zeros
124 integer variables, 6 of which are binary
Scaling...
 A: min|aij| =  5.034e-01  max|aij| =  1.000e+02  ratio =  1.986e+02
GM: min|aij| =  5.042e-01  max|aij| =  1.983e+00  ratio =  3.933e+00
EQ: min|aij| =  2.601e-01  max|aij| =  1.000e+00  ratio =  3.845e+00
2N: min|aij| =  2.123e-01  max|aij| =  1.219e+00  ratio =  5.742e+00
Constructing initial basis...
Size of triangular part is 52
Solving LP relaxation...
GLPK Simplex Optimizer 5.0
52 rows, 124 columns, 786 non-zeros
      0: obj =   1.478120997e+05 inf =   3.374e+02 (4)
     14: obj =   1.401812915e+05 inf =   0.000e+00 (0)
*    55: obj =   9.442241382e+04 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Integer optimization begins...
Long-step dual simplex will be used
+    55: mip =     not found yet >=              -inf        (1; 0)
+    58: >>>>>   9.442241382e+04 >=   9.442241382e+04   0.0% (4; 0)
+    58: mip =   9.442241382e+04 >=     tree is empty   0.0% (0; 7)
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.3 Mb (271682 bytes)
Writing MIP solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmpdh24sp6z.glpk.raw'...
187 lines were written
Instance with 18 payloads, 6 rockets, and 2 destinations:
Optimal Payload Assignments: {(0, 0): 0.0, (0, 1): 0.0, (0, 2): 0.0, (0, 3): 12.0, (0, 4): 0.0, (0, 5): 0.0, (1, 0): 0.0, (1, 1): 0.0, (1,
2): 6.0, (1, 3): 0.0, (1, 4): 0.0, (1, 5): 0.0, (2, 0): 0.0, (2, 1): 0.0, (2, 2): 5.0, (2, 3): 0.0, (2, 4): 0.0, (2, 5): 0.0, (3, 0): 0.0,
(3, 1): 6.0, (3, 2): 0.0, (3, 3): 0.0, (3, 4): 0.0, (3, 5): 0.0, (4, 0): 0.0, (4, 1): 0.0, (4, 2): 0.0, (4, 3): 14.0, (4, 4): 0.0, (4, 5):
0.0, (5, 0): 0.0, (5, 1): 0.0, (5, 2): 5.0, (5, 3): 0.0, (5, 4): 0.0, (5, 5): 0.0, (6, 0): 0.0, (6, 1): 0.0, (6, 2): 1.0, (6, 3): 14.0, (6,
4): 0.0, (6, 5): 0.0, (7, 0): 0.0, (7, 1): 0.0, (7, 2): 6.0, (7, 3): 0.0, (7, 4): 0.0, (7, 5): 0.0, (8, 0): 0.0, (8, 1): 6.0, (8, 2): 0.0,
(8, 3): 0.0, (8, 4): 0.0, (8, 5): 0.0, (9, 0): 0.0, (9, 1): 0.0, (9, 2): 0.0, (9, 3): 10.0, (9, 4): 0.0, (9, 5): 0.0, (10, 0): 0.0, (10,
1): 8.0, (10, 2): 0.0, (10, 3): 0.0, (10, 4): 0.0, (10, 5): 0.0, (11, 0): 0.0, (11, 1): 0.0, (11, 2): 0.0, (11, 3): 5.0, (11, 4): 0.0, (11,
5): 0.0, (12, 0): 0.0, (12, 1): 0.0, (12, 2): 5.0, (12, 3): 0.0, (12, 4): 0.0, (12, 5): 0.0, (13, 0): 0.0, (13, 1): 0.0, (13, 2): 5.0, (13,
3): 0.0, (13, 4): 0.0, (13, 5): 0.0, (14, 0): 0.0, (14, 1): 10.0, (14, 2): 0.0, (14, 3): 0.0, (14, 4): 0.0, (14, 5): 0.0, (15, 0): 0.0, (1
5, 1): 15.0, (15, 2): 0.0, (15, 3): 0.0, (15, 4): 0.0, (15, 5): 0.0, (16, 0): 0.0, (16, 1): 0.0, (16, 2): 14.0, (16, 3): 0.0, (16, 4): 0.0,
(16, 5): 0.0, (17, 0): 0.0, (17, 1): 0.0, (17, 2): 12.0, (17, 3): 0.0, (17, 4): 0.0, (17, 5): 0.0}
Total Weight on Rockets: {0: 0.0, 1: 45.0, 2: 59.0, 3: 55.0, 4: 0.0, 5: 0.0}
Weight Assigned to Destinations: {0: 159.0, 1: 159.0}
Optimal Cost: 94422.41381868944
----------------------------------------
```

In [41]:

```python
random.seed(23)
solve_instance()
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmpl77b3ce2.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmpqh7haaxu.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmpialylto4.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmpialylto4.pyomo.lp'...
C:\Users\Gabriel\AppData\Local\Temp\tmpialylto4.pyomo.lp:1336: warning: lower bound of variable 'x119' redefined
C:\Users\Gabriel\AppData\Local\Temp\tmpialylto4.pyomo.lp:1336: warning: upper bound of variable 'x119' redefined
53 rows, 125 columns, 807 non-zeros
124 integer variables, 4 of which are binary
1340 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmpqh7haaxu.glpk.glp'...
1272 lines were written
GLPK Integer Optimizer 5.0
53 rows, 125 columns, 807 non-zeros
124 integer variables, 4 of which are binary
Preprocessing...
52 rows, 124 columns, 806 non-zeros
124 integer variables, 4 of which are binary
Scaling...
 A: min|aij| =  5.049e-01  max|aij| =  1.000e+02  ratio =  1.981e+02
GM: min|aij| =  4.344e-01  max|aij| =  2.302e+00  ratio =  5.299e+00
EQ: min|aij| =  2.031e-01  max|aij| =  1.000e+00  ratio =  4.923e+00
2N: min|aij| =  1.250e-01  max|aij| =  1.376e+00  ratio =  1.101e+01
Constructing initial basis...
Size of triangular part is 52
Solving LP relaxation...
GLPK Simplex Optimizer 5.0
52 rows, 124 columns, 806 non-zeros
      0: obj =   1.694489034e+05 inf =   7.783e+02 (5)
     32: obj =   2.059175795e+05 inf =   0.000e+00 (0)
*    53: obj =   1.730913052e+05 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Integer optimization begins...
Long-step dual simplex will be used
+    53: mip =     not found yet >=              -inf        (1; 0)
Solution found by heuristic: 173091.305176
+    54: mip =   1.730913052e+05 >=     tree is empty   0.0% (0; 1)
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.3 Mb (262630 bytes)
Writing MIP solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmpl77b3ce2.glpk.raw'...
187 lines were written
Instance with 28 payloads, 4 rockets, and 2 destinations:
Optimal Payload Assignments: {(0, 0): 12.0, (0, 1): 0.0, (0, 2): 0.0, (0, 3): 0.0, (1, 0): 0.0, (1, 1): 0.0, (1, 2): 6.0, (1, 3): 0.0, (2,
0): 0.0, (2, 1): 0.0, (2, 2): 5.0, (2, 3): 0.0, (3, 0): 0.0, (3, 1): 6.0, (3, 2): 0.0, (3, 3): 0.0, (4, 0): 0.0, (4, 1): 0.0, (4, 2): 14.0,
(4, 3): 0.0, (5, 0): 0.0, (5, 1): 0.0, (5, 2): 0.0, (5, 3): 5.0, (6, 0): 0.0, (6, 1): 15.0, (6, 2): 0.0, (6, 3): 0.0, (7, 0): 0.0, (7, 1):
0.0, (7, 2): 6.0, (7, 3): 0.0, (8, 0): 0.0, (8, 1): 0.0, (8, 2): 6.0, (8, 3): 0.0, (9, 0): 0.0, (9, 1): 0.0, (9, 2): 0.0, (9, 3): 10.0, (1
0, 0): 8.0, (10, 1): 0.0, (10, 2): 0.0, (10, 3): 0.0, (11, 0): 3.0, (11, 1): 0.0, (11, 2): 0.0, (11, 3): 2.0, (12, 0): 0.0, (12, 1): 0.0,
(12, 2): 5.0, (12, 3): 0.0, (13, 0): 0.0, (13, 1): 5.0, (13, 2): 0.0, (13, 3): 0.0, (14, 0): 0.0, (14, 1): 0.0, (14, 2): 0.0, (14, 3): 10.
0, (15, 0): 15.0, (15, 1): 0.0, (15, 2): 0.0, (15, 3): 0.0, (16, 0): 0.0, (16, 1): 0.0, (16, 2): 14.0, (16, 3): 0.0, (17, 0): 0.0, (17, 1):
0.0, (17, 2): 12.0, (17, 3): 0.0, (18, 0): 9.0, (18, 1): 0.0, (18, 2): 5.0, (18, 3): 0.0, (19, 0): 0.0, (19, 1): 0.0, (19, 2): 0.0, (19,
3): 12.0, (20, 0): 0.0, (20, 1): 0.0, (20, 2): 0.0, (20, 3): 7.0, (21, 0): 0.0, (21, 1): 0.0, (21, 2): 0.0, (21, 3): 6.0, (22, 0): 0.0, (2
2, 1): 0.0, (22, 2): 0.0, (22, 3): 7.0, (23, 0): 0.0, (23, 1): 0.0, (23, 2): 6.0, (23, 3): 0.0, (24, 0): 0.0, (24, 1): 0.0, (24, 2): 5.0,
(24, 3): 0.0, (25, 0): 7.0, (25, 1): 6.0, (25, 2): 0.0, (25, 3): 0.0, (26, 0): 0.0, (26, 1): 0.0, (26, 2): 0.0, (26, 3): 12.0, (27, 0): 0.
0, (27, 1): 0.0, (27, 2): 0.0, (27, 3): 8.0}
Total Weight on Rockets: {0: 54.0, 1: 32.0, 2: 84.0, 3: 79.0}
Weight Assigned to Destinations: {0: 249.0, 1: 249.0}
Optimal Cost: 173091.30517581088
----------------------------------------
```

In [39]:

```python
random.seed(33)
solve_instance()
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write C:\Users\Gabriel\AppData\Local\Temp\tmpp_u70mf4.glpk.raw --wglp C:\Users\Gabriel\AppData\Local\Temp\tmpkum_zk_u.glpk.glp
 --cpxlp C:\Users\Gabriel\AppData\Local\Temp\tmpbww8egk8.pyomo.lp
Reading problem data from 'C:\Users\Gabriel\AppData\Local\Temp\tmpbww8egk8.pyomo.lp'...
C:\Users\Gabriel\AppData\Local\Temp\tmpbww8egk8.pyomo.lp:3322: warning: lower bound of variable 'x241' redefined
C:\Users\Gabriel\AppData\Local\Temp\tmpbww8egk8.pyomo.lp:3322: warning: upper bound of variable 'x241' redefined
82 rows, 251 columns, 2336 non-zeros
250 integer variables, 5 of which are binary
3327 lines were read
Writing problem data to 'C:\Users\Gabriel\AppData\Local\Temp\tmpkum_zk_u.glpk.glp'...
3225 lines were written
GLPK Integer Optimizer 5.0
82 rows, 251 columns, 2336 non-zeros
250 integer variables, 5 of which are binary
Preprocessing...
81 rows, 250 columns, 2335 non-zeros
250 integer variables, 5 of which are binary
Scaling...
 A: min|aij| =  5.792e-01  max|aij| =  1.000e+02  ratio =  1.727e+02
GM: min|aij| =  4.494e-01  max|aij| =  2.225e+00  ratio =  4.952e+00
EQ: min|aij| =  2.157e-01  max|aij| =  1.000e+00  ratio =  4.635e+00
2N: min|aij| =  1.250e-01  max|aij| =  1.304e+00  ratio =  1.044e+01
Constructing initial basis...
Size of triangular part is 81
Solving LP relaxation...
GLPK Simplex Optimizer 5.0
81 rows, 250 columns, 2335 non-zeros
      0: obj =   3.507863418e+05 inf =   1.465e+03 (4)
     70: obj =   3.041052300e+05 inf =   1.350e+02 (2)
LP HAS NO PRIMAL FEASIBLE SOLUTION
Time used:   0.0 secs
Memory used: 0.5 Mb (542892 bytes)
Writing MIP solution to 'C:\Users\Gabriel\AppData\Local\Temp\tmpp_u70mf4.glpk.raw'...
342 lines were written
Instance with 46 payloads, 5 rockets, and 5 destinations is infeasible.
----------------------------------------
```

Solution Comparison and Analysis

The time required to solve integer programming problems with GLPK increases with the size and complexity of the problem, especially as the number of integer and binary variables grows. While small and medium instances can be solved quickly, larger instances may lead to infeasibility or significantly increased solution times due to the expanded solution space and stricter constraints.

Instance 1: Small to Medium Size

- **Problem Size**: 18 payloads, 6 rockets, 2 destinations
- **Solution**:
  - **Status**: Feasible and optimal
  - **Optimal Cost**: 94,422.41
  - **Time Required**: 0.0 seconds

This instance, being relatively small, was solved quickly by GLPK with no delay. The optimization was able to find an integer optimal solution with minimal computational effort, likely due to the limited number of payloads, rockets, and destinations, resulting in a smaller solution space.

Instance 2: Medium Size

- **Problem Size**: 28 payloads, 4 rockets, 2 destinations
- **Model Statistics**:
  - Rows: 53
  - Columns: 125
  - Non-zeros: 807
  - Integer variables: 124 (4 binary)
- **Solution**:
  - **Status**: Feasible and optimal
  - **Optimal Cost**: 173,091.31
  - **Time Required**: 0.0 seconds

This instance is slightly larger than the first in terms of payloads, but with fewer rockets. While the number of variables is similar, the slightly more complex assignment of payloads resulted in a higher optimal cost. However, this instance was also solved rapidly, suggesting that GLPK can handle medium-sized instances with this structure efficiently when they are feasible.

Instance 3: Larger and More Complex Size

- **Problem Size**: 46 payloads, 5 rockets, 5 destinations
- **Model Statistics**:
  - Rows: 82
  - Columns: 251
  - Non-zeros: 2336
  - Integer variables: 250 (5 binary)
- **Solution**:
  - **Status**: Infeasible
  - **Time Required**: 0.0 seconds (reported)

This larger instance, with more payloads, rockets, and destinations, led to an infeasible solution. The increased complexity, reflected by the higher number of rows, columns, and non-zero coefficients, likely introduced constraints that were impossible to satisfy simultaneously. Although GLPK quickly reported the infeasibility, solving larger instances or identifying infeasibility in such cases can be challenging due to the expanded solution space.

Problem Size and Solution Time

1. **Scalability**:
   - The GLPK solver handled the smaller and medium instances with ease, reporting optimal solutions in negligible time. This is expected, as instances with fewer variables and constraints are typically less complex.
   - As the number of payloads, rockets, and destinations increased, the model grew significantly in complexity, as seen in the third instance with 82 rows and 251 columns.
2. **Impact of Larger Instances**:
   - The third instance highlights that increasing the number of payloads, rockets, and destinations can lead to infeasibility or longer solution times. This is particularly noticeable when the additional constraints (more destinations and rockets) add restrictions that are difficult to satisfy together.
   - In larger instances, even if feasible, the time required to explore and prune the expanded solution space in integer programming can increase significantly. Although GLPK reported infeasibility quickly in this example, complex feasible cases might require more time.
3. **Memory and Computational Resources**:
   - As the size of the problem increases, the memory usage also grows. For instance, the first instance used approximately 0.3 MB, whereas the larger, infeasible instance required about 0.5 MB.
   - This increased memory requirement reflects the larger data structures needed to store and manipulate more variables and constraints.

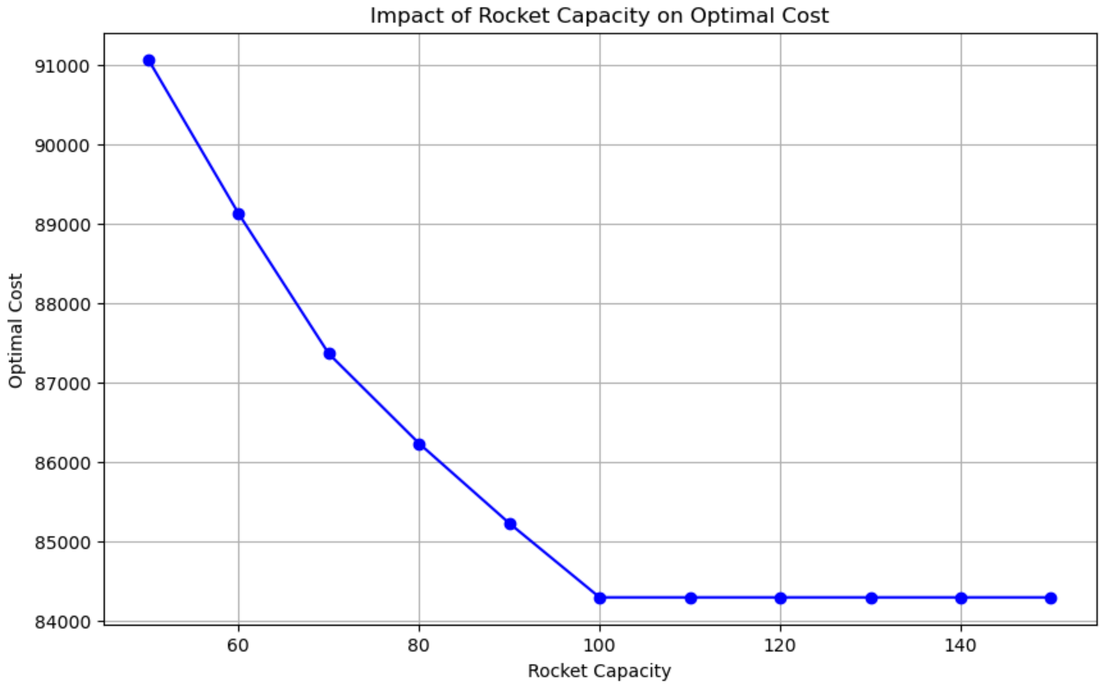
h) Rocket Capacity variations

To gain insights into how the model's objective function changes with varying parameters, we could focus on the **rocket** capacity parameter. Changing the capacity of each rocket should impact the solution because it directly affects the amount of payload each rocket can carry, which could lead to changes in payload assignments, rocket activations, and ultimately the cost.

In [42]:
```
from pyomo.environ import *
import random
import matplotlib.pyplot as plt

# Seed for reproducibility
random.seed(1234)

# Define a function to solve the problem with varying rocket capacities
def solve_instance_with_capacity(rocket_capacity_values):
    results = []

    # Fixed values for other parameters
    num_payloads = 20
    num_rockets = 5
    num_destinations = 3
    weights = [random.randint(5, 15) for _ in range(num_payloads)]  # Random weights for each payload
    fuel_capacity = [300] * num_rockets  # Fixed fuel capacity for all rockets
    launch_costs = [random.randint(400, 1000) for _ in range(num_rockets)]  # Random launch cost per rocket
    distances = [random.randint(1, 5) for _ in range(num_destinations)]  # Random distance to each destination
    fuel_requirements = [[random.uniform(0.5, 2.5) * distances[random.randint(0, num_destinations-1)]
                          for _ in range(num_rockets)] for _ in range(num_payloads)]

    # Loop over different rocket capacities
    for rocket_capacity_value in rocket_capacity_values:
        # Initialize the Pyomo model
        model = ConcreteModel()

        # Sets
        N = num_payloads       # Number of payloads
        K = num_rockets        # Number of rockets
        M = num_destinations   # Number of destinations

        # Variable rocket capacities
        rocket_capacity = [rocket_capacity_value] * K  # Apply the current capacity value to all rockets

        # Decision Variables
        model.x = Var(range(N), range(K), domain=NonNegativeIntegers)  # Payload assignment (weight)
        model.y = Var(range(K), domain=NonNegativeIntegers)            # Total weight on each rocket
        model.z = Var(range(M), domain=NonNegativeIntegers)            # Total weight to each destination
        model.b = Var(range(K), domain=Binary)                         # Binary variable for rocket activation
        model.p = Var(range(M), domain=NonNegativeIntegers)            # Priority levels for destinations

        # Objective Function: Minimize total cost
        model.obj = Objective(
            expr=sum(launch_costs[k] * model.y[k] for k in range(K)) +
                 sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N) for k in range(K)),
            sense=minimize
        )

        # Constraints
        model.payload_assignment = ConstraintList()
        for n in range(N):
            model.payload_assignment.add(sum(model.x[n, k] for k in range(K)) == weights[n])

        model.rocket_capacity = ConstraintList()
        for k in range(K):
            model.rocket_capacity.add(sum(model.x[n, k] for n in range(N)) <= rocket_capacity[k])

        model.fuel_capacity = ConstraintList()
        for k in range(K):
            model.fuel_capacity.add(sum(fuel_requirements[n][k] * model.x[n, k] for n in range(N)) <= fuel_capacity[k])

        model.rocket_weight_consistency = ConstraintList()
        for k in range(K):
            model.rocket_weight_consistency.add(model.y[k] == sum(model.x[n, k] for n in range(N)))

        M_large = 100
        model.rocket_activation = ConstraintList()
        for k in range(K):
            model.rocket_activation.add(sum(model.x[n, k] for n in range(N)) <= M_large * model.b[k])

        W_min = 15
        model.min_payload_requirement = ConstraintList()
        for k in range(K):
            model.min_payload_requirement.add(model.y[k] >= W_min * model.b[k])

        P_min = 30
        model.priority_assignment = ConstraintList()
        for m in range(M):
            model.priority_assignment.add(model.z[m] >= P_min * model.p[m])
            model.priority_assignment.add(model.z[m] == sum(model.x[n, k] for n in range(N) for k in range(K) if distances[m]))

        # Solve the model using GLPK solver
        solver = SolverFactory('glpk')
        results_obj = solver.solve(model, tee=False)

        # Check if a feasible solution was found. As before, this is to avoid errors
        if (results_obj.solver.termination_condition == TerminationCondition.optimal or
            results_obj.solver.termination_condition == TerminationCondition.feasible):
            optimal_cost = model.obj()
            results.append((rocket_capacity_value, optimal_cost))
            print(f"Rocket capacity: {rocket_capacity_value}, Optimal Cost: {optimal_cost}")
        else:
            print(f"Rocket capacity: {rocket_capacity_value} led to an infeasible solution.")
            results.append((rocket_capacity_value, None))
```

```
    return results
```

```
Rocket capacity: 50, Optimal Cost: 91071.00195026731
Rocket capacity: 60, Optimal Cost: 89130.76066811134
Rocket capacity: 70, Optimal Cost: 87365.98061152514
Rocket capacity: 80, Optimal Cost: 86236.79808203693
Rocket capacity: 90, Optimal Cost: 85231.58731496906
Rocket capacity: 100, Optimal Cost: 84298.56475375145
Rocket capacity: 110, Optimal Cost: 84298.56475375145
Rocket capacity: 120, Optimal Cost: 84298.56475375145
Rocket capacity: 130, Optimal Cost: 84298.56475375145
Rocket capacity: 140, Optimal Cost: 84298.56475375145
Rocket capacity: 150, Optimal Cost: 84298.56475375145
```



In [ ]:
```
# Define the range of rocket capacities to test
rocket_capacity_values = list(range(50, 151, 10))  # 10 values between 50 and 150

# Run the experiment and collect results
results = solve_instance_with_capacity(rocket_capacity_values)
```

In [ ]:
```
# Extract values for plotting
capacity_values = [r[0] for r in results if r[1] is not None]
optimal_costs = [r[1] for r in results if r[1] is not None]

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(capacity_values, optimal_costs, marker='o', linestyle='-', color='b')
plt.xlabel("Rocket Capacity")
plt.ylabel("Optimal Cost")
plt.title("Impact of Rocket Capacity on Optimal Cost")
plt.grid(True)
plt.show()
```

Analysis of Rocket Capacity Impact on Optimal Cost

The results demonstrate that increasing rocket capacity has a significant impact on reducing the optimal cost up to a certain threshold, after which the cost stabilizes. This behavior highlights the importance of balancing capacity with operational costs. For the given data and constraints, a rocket capacity of 100 provides the best cost-efficiency, beyond which additional capacity does not lead to further cost reductions.

1. **Cost Decrease with Increasing Capacity**:
   - As shown in the graph, the **optimal cost decreases as the rocket capacity increases**. This trend is particularly noticeable for capacities between 50 and 100, where the cost reduction is significant. For example:
     - At a rocket capacity of 50, the optimal cost is approximately 91,071.
     - At a capacity of 100, the optimal cost reduces to approximately 84,298.
2. **Cost Stabilization Beyond Capacity of 100**:
   - After the rocket capacity reaches 100, increasing the capacity further has no noticeable effect on the optimal cost, which stabilizes at around 84,298. This suggests that a capacity of 100 is sufficient to achieve an optimal configuration that minimizes the cost effectively.
3. **Capacity Threshold Effect**:
   - The stabilization of the optimal cost beyond a capacity of 100 suggests a **threshold effect**. Up to this threshold, increasing capacity allows for more efficient payload assignments, reducing the need for additional rockets or lowering fuel costs. However, beyond this threshold, additional capacity does not yield further improvements, likely because the problem constraints are already satisfied optimally at this level.

Interpretation

1. **Impact of Rocket Capacity on Payload Assignment**:
    - At lower capacities, rockets are limited in the amount of payload they can carry, potentially requiring more rockets to be activated or leading to suboptimal assignments that increase fuel and launch costs. As capacity increases, payloads can be distributed more efficiently, reducing the need for additional rocket activations and lowering costs.
2. **Optimal Cost Saturation**:
    - Once the rocket capacity reaches 100, further increases no longer improve the solution. This suggests that, with the current payload and destination requirements, a capacity of 100 per rocket is sufficient to fully meet all constraints in an optimal manner. This capacity level likely enables payloads to be assigned in a way that minimizes both the launch and fuel costs.
3. **Practical Implications**:
    - This analysis provides insights for decision-making regarding rocket specifications. It indicates that, for the given problem structure, a rocket capacity of 100 is optimal and further increases would not contribute to additional cost savings. This insight can be valuable for budgeting and resource allocation in scenarios where rocket capacities are adjustable.