Modelowanie Matematyczne, Projekt 3, Dane nr 1.6

Krzysztof Rudnicki, 307585

23 stycznia 2024

0 Wstęp

2 Fabryki, F1, F2
4 Magazyny, M1, M2, M3, M4
6 Klientów, K1, K2, K3, K4, K5, K6

Koszty dystrybucji towaru

zaopatruje	F1	F2	M1	M2	M3	M4
Magazyny						
M1	0.3	-				
M2	0.5	0.4				
M3	1.2	0.5				
M4	0.8	0.3				
Klientów						
K1	1.2	1.8	-	1.4	-	-
K2	-	-	1.2	0.3	1.3	-
K3	1.2	-	0.2	1.8	2.0	0.4
K4	2.0	-	1.7	1.3	-	2.0
K5	-	-	-	0.5	0.3	0.5
K6	1.1	-	2.0	-	1.4	1.5

1 Model Dwukryterialny

Zbiory

- $i, k \in F$ - Fabryki
- $i, l \in M$ - Magazyny
- $j \in K$ - Klienci

Parametery

- $C_{i,j}$ - Koszty transportu dóbr z punktów i (fabryka lub magazyn) do klienta j
- $C_{k,l}$ - Koszty transportu dóbr z fabryki i do magazynu k
- $P_{i,j}$ - Binarnie określa czy preferencja klienta zostąła spełniona (1) czy nie (0)
- S_j - Poziom satysfakcji klienta j
- D_j - Zapotrzebowanie klienta j

Zmienne decyzyjne

- $x_{i,j}$ - Liczba dóbr (w tys. ton) przetransportowana z punktu i (fabryka lub magazyn) do klienta j
- $y_{k,l}$ - Liczba dóbr (w tys. ton) przetransportowana z fabryki k do magazynu l

Funkcja celu

1. Minimalizacja kosztów dystrybucji

$$Min(\sum_{i,j} C_{i,j} * x_{i,j} + \sum i, kC_{i,k} * y_{i,k})$$

2. Maksymalizacja satysfakcja klienta
W celu maksymalizacji satysfakcji klienta policzymy ile z dostaw do klientów odbyło się z preferencyjnych źródeł

$$Max(\sum_{j} S_j * P_{i,j} * x_{i,j})$$

Ograniczenia Miesięczne możliwości produkcyjne fabryk

$$\sum_j x_{F1,j} + \sum_k y_{F1,k} \leq 150 \tag{1}$$

$$\sum_j x_{F2,j} + \sum_k y_{F2,k} \leq 200 \tag{2}$$

Miesięczna ilośc obsługiwanego towaru przez magazyny

$$\sum_j x_{M1,j} \leq 70 \tag{3}$$

$$\sum_j x_{M2,j} \leq 50 \tag{4}$$

$$\sum_j x_{M3,j} \leq 100 \tag{5}$$

$$\sum_j x_{M4,j} \leq 40 \tag{6}$$

Spełnienie preferencji klienta

$$\sum_i x_{i,j} = D_j \tag{7}$$

Wartości niezerowe

$$x_{i,j}, y_{i,k} \geq 0 \tag{8}$$

2 Implementacja

Do implementacji użyty został python z biblioteką pulp https://coin-or.github.io/pulp/index.html Dzięki temu wykorzystujemy zarówno łatwość pythona jak i możliwości używania różnych solverów (CBC, GLPK, CPLEX, Gurobi...) przez pulpa

Listing 1: Import bilbioteki pulp

```
import pulp
```

Listing 2: Iniicjalizacja modelu

```
model = pulp.LpProblem("Optimal_Distribution", pulp.LpMinimize)
```

Listing 3: Zmienne decyzyjne

```
x =
pulp.LpVariable.dicts("x",
[(i, j) for i in punkty for j in klienci],
lowBound=0,
cat='Integer')
y
 pulp.LpVariable.dicts("y",
 [(i, k) for i in fabryki for k in magazyny],
 lowBound=0,
 cat='Integer')
```

Listing 4: Funkcje celu

```
# Objective function components
koszt_dystrybucji =
pulp.lpSum(
    [cost[i][j] * x[(i, j)] for i in punkty for j in klienci]
    )
koszt_magazynowania =
pulp.lpSum(
    [cost[i][k] * y[(i, k)] for i in fabryki for k in magazyny]
    )

# Define objective
model +=
alpha
* (koszt_dystrybucji + koszt_magazynowania)
- beta * poziom_satysfakcji
```

Listing 5: Ograniczenia

```
for i in fabryki:
    model +=
        pulp.lpSum([x[(i, j)] for j in klienci]
        + [y[(i, k)] for k in magazyny])
        <= mozliwosci_fabryki[i]
for k in magazyny:
    model +=
```

```
    pulp.lpSum([x[(k, j)] for j in klienci])
    <= mozliwosci_magazynu[k]
for j in klienci:
    model +=
    pulp.lpSum([x[(i, j)] for i in punkty]) == wymagania_klienta[j]
```

Listing 6: Rozwiązanie problemu

```
    model.solve()
```

Listing 7: Przedstawienie wyników

```
for v in model.variables():
    print(v.name, "=", v.varValue)
```

3 Rozwiązanie efektywne

Aby zdefiniować rozwiązanie efektywne sprawdzamy sumaryczy obu funkcji celu dla wartości α i *beta* od 1 do 10
w tym celu napisany został kod który modyfikuje wartości α i β w pętli

Listing 8: Wyznaczanie alpha i beta

```
# Varying alpha and beta
for alpha in range(0, 11):
    beta = 10 - alpha
    alpha /= 10.0
    beta /= 10.0

    # Update objective function
    model.objective = alpha * cost_distribution - beta * satisfaction_co

    # Solve the model
    model.solve()

    # Record the results
    total_cost = value(cost_distribution)
    total_satisfaction = value(satisfaction_component)
    cost_results.append(total_cost)
```

```
satisfaction_results.append(total_satisfaction)
```