Min Cost Flow Problem

Μοντελοποίηση και Επίλυση με Τεχνικές Γραμμικού Προγραμματισμού

Γεώργιος Παπουτσάς

AM:1083738

Έτος: 4^{o}

Επισκόπηση του Προβλήματος

Το πρόβλημα Ελαχίστου Κόστους Ροής (MCF) είναι ένα θεμελιώδες πρόβλημα στη βελτιστοποίηση δικτύων, το οποίο αφορά στην εξεύρεση της λιγότερο δαπανηρής μεθόδου μεταφοράς μιας συγκεκριμένης ποσότητας ροής μέσω ενός δικτύου από κόμβους προμήθειας (πηγές) προς κόμβους ζήτησης (καταβόθρες). Το δίκτυο αναπαρίσταται ως ένας κατευθυνόμενος Γράφος, όπου κάθε ακμή (ή τόξο) έχει ένα σχετικό κόστος ανά μονάδα ροής και μια χωρητικότητα που περιορίζει τη μέγιστη ροή που μπορεί να μεταφέρει. Ο στόχος του προβλήματος MCF είναι να προσδιορίσει τη βέλτιστη ροή μέσω του δικτύου που ικανοποιεί τους περιορισμούς προμήθειας και ζήτησης με το ελάχιστο δυνατό συνολικό κόστος.

Μοντελοποίηση Απλών Δικτύων

- Το πρόβλημα διατυπώνεται με ένα γράφο: $G = (V, E)$
- Ροές: $f(u, v)$
- Κόστη: $a(u, v)$
- Χωρητικότητες: $c(u, v)$
- Προσφορά ή ζήτηση κάθε κόμβου: $s(i)$

Όπου

$$i \in V$$

$$(u, v) \in E$$

Μεταβλητές Απόφασης και Αντικειμενική Συνάρτηση

- Οι ροές των κάθε ακμών αποτελούν τις μεταβλητές απόφασης
- Αντικειμενική Συνάρτηση:

$$\sum_{(u,v)\in E} a(u,v) \cdot f(u,v)$$

Περιορισμοί

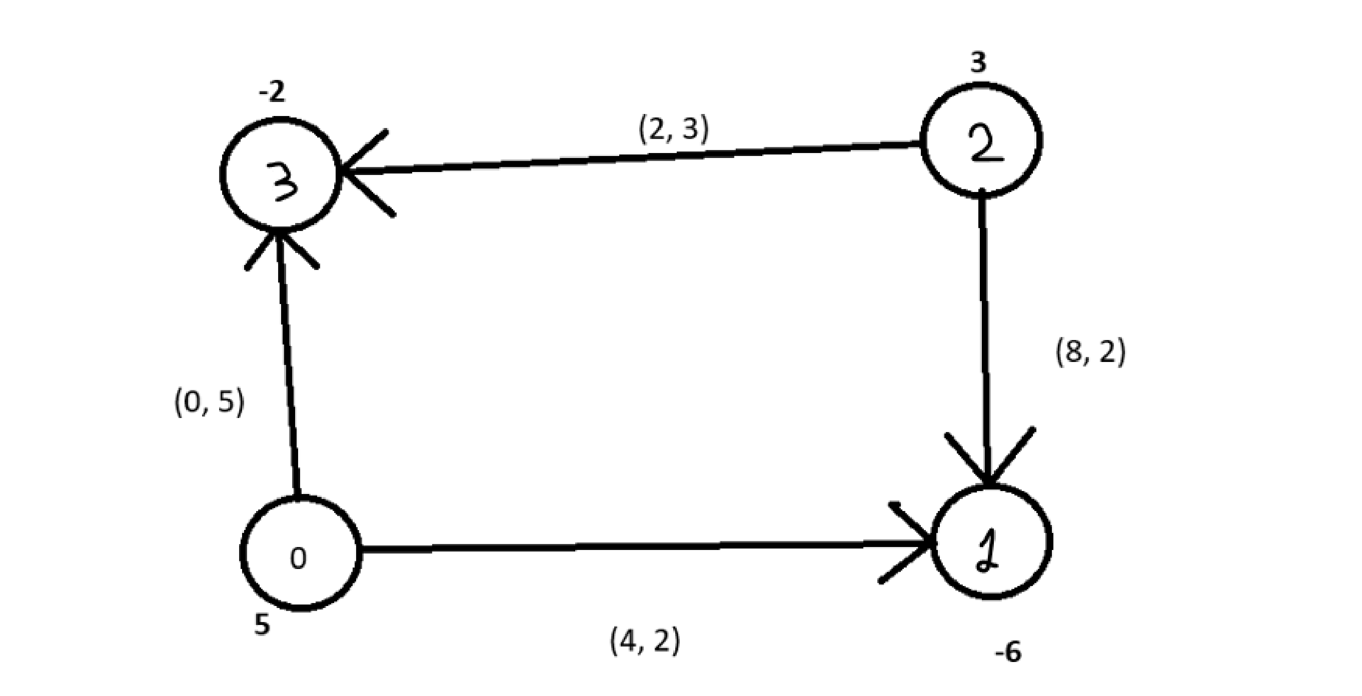
- Προσφορά και ζήτηση καλύπτονται για όλους τους κόμβους:
  - $\sum_{j:(i,j)\in E} f(i,j) - \sum_{k:(k,i)\in E} f(k,i) = s(i) \ \gamma\iota\alpha \ \kappa\alpha\theta\varepsilon \ i \in V$

- Χωρητικότητες ακμών ικανοποιούνται:
  - $f(i,j) \leq c(i,j) \ \gamma\iota\alpha \ \kappa\alpha\theta\varepsilon \ (i,j) \in E$
- Οι ροές σε κάθε ακμή είναι μεγαλύτερες ή ίσες με 0:
  - $f(i,j) \geq 0 \ \gamma\iota\alpha \ \kappa\alpha\theta\varepsilon \ (i,j) \in E$

Δεδομένα Εισόδου/Εξόδου υλοποίησης σε Python

- Είσοδοι:
  - Γράφος δικτύου (κόμβοι, ακμές)
  - Κόστος κάθε ακμής
  - Χωρητικότητα κάθε ακμής
  - Προσφορά ή Ζήτηση κάθε κόμβου
- Έξοδος:
  - Το ελάχιστο κόστος
  - Ροές σε κάθε ακμή

Παράδειγμα Απλού Δικτύου



Μοντελοποίηση του Απλού Δικτύου σε Γραμμικό Πρόγραμμα

$$\min Z = 0x_1 + 4x_2 + 2x_3 + 8x_4$$

Υπο τους περιορισμούς:

$$x_1 + x_2 = 5$$

$$x_1 + x_3 = 6$$

$$x_3 + x_4 = 3$$

$$-x_2 + x_4 = -2$$

$$0 \leq x_1 \leq 5$$

$$0 \leq x_2 \leq 2$$

$$0 \leq x_3 \leq 3$$

$$0 \leq x_2 \leq 2$$

Λύση του Απλού Δικτύου

- Χρησιμοποιώντας μεθόδους Simplex και συγκεκριμένα για προβλήματα με ακέραιες τιμές παίρνουμε την εξής λύση:
  - $x_1 = 3, x_2 = 2, x_3 = 3, x_4 = 0$
  - $Z = 14$
- Δηλαδή η ροή από $0 \rightarrow 1$ θα είναι 3, από $0 \rightarrow 3$ θα είναι 2, από $2 \rightarrow 1$ θα είναι 3 και από $2 \rightarrow 3$ θα είναι 0 και το ελάχιστο κόστος θα είναι 14.

Μοντελοποίηση Πραγματικού Προβλήματος Crown Distributors Company Kegalle, Sri Lanka

- Η εταιρεία έχει δύο εργοστάσια(P_1, P_2), τέσσερεις αποθήκες (W_1, W_2, W_3, W_4) και πουλάει τα προϊόντα της σε έξι διαφορετικούς πελάτες ($C_1, C_2, C_3, C_4, C_5, C_6$). Τα κόστη διανομής είναι γνωστά και παρατίθενται στον πίνακα της επόμενης διαφάνειας. Η παύλα σημαίνει αδυναμία αποστολής από κάποιον προμηθευτή (εργοστάσιο ή αποθήκη) προς κάποια αποθήκη ή πελάτη και το κενό σημαίνει ότι μια αποθήκη δεν μπορεί να προμηθεύσει τον εαυτό της (προφανώς) ή άλλες αποθήκες.

Κόστη διανομής

Προμηθεύει	Προμηθευτής					
	P_1	P_2	W_1	W_2	W_3	W_4
W_1	0.5	-				
W_2	0.5	0.3				
W_3	1.0	0.5				
W_4	0.2	0.2				
C_1	1.0	2.0	-	1.0	-	-
C_2	-	-	1.5	0.5	1.5	-
C_3	1.5	-	0.5	0.5	2.0	0.2
C_4	2.0	-	1.5	1.0	-	1.5
C_5	-	-	-	0.5	0.5	0.5
C_6	1.0	-	1.0	-	1.5	1.5

Χωρητικότητες Εργοστασίων, Αποθηκών, Απαιτήσεις Πελατών

- P_1 150000
- P_2 200000

- W_1 70000
- W_2 50000
- W_3 100000
- W_4 40000

- C_1 50000
- C_2 10000
- C_3 40000
- C_4 35000
- C_5 60000
- C_6 20000

Μεταβλητές Απόφασης και Αντικειμενική Συνάρτηση

- x_{ij} : ποσότητα που στέλνεται από το εργοστάσιο i προς την αποθήκη j
  - $i = 1, 2, \; j = 1, 2, 3, 4$
- y_{ik} : ποσότητα που στέλνεται από το εργοστάσιο i προς τον πελάτη k
  - $i = 1, 2, \; k = 1, 2, 3, 4, 5, 6$
- z_{jk} : ποσότητα που στέλνεται από την αποθήκη j προς τον πελάτη k
  - $j = 1, 2, 3, 4, \quad k = 1, 2, 3, 4, 5, 6$
- 29 συνολικά

- $\sum_{i=1,j=1}^{i=2,j=4} c_{ij}x_{ij} + \sum_{i=1,k=1}^{i=2,k=6} d_{ik}y_{ik} + \sum_{j=1,k=1}^{j=4,k=6} e_{jk}z_{jk}$

Περιορισμοί

1. Περιορισμοί Εργοστασίων:
    - $\sum_{j=1}^{2} x_{ij} + \sum_{k=1}^{6} y_{ik} \leq χωρητικότητα[i],\ i = 1, 2$
1. Ποσότητες προς αποθήκες:
    - $\sum_{i=1}^{2} x_{ij} \leq παροχη[j],\ j = 1, 2, 3, 4$
1. Ποσότητες από αποθήκες:
    - $\sum_{k=1}^{6} z_{jk} = \sum_{i=1}^{2} x_{ij}\ ,\ j = 1, 2, 3, 4$
1. Απαιτήσεις πελατών:
    - $\sum_{i=1}^{2} y_{ik} + \sum_{j=1}^{4} z_{jk} = απαιτηση[k],\ k = 1, 2, 3, 4, 5, 6$

Δεδομένα Εισόδου/Εξόδου υλοποίησης σε Python

```python
def min_cost_flow_ilp_factory(nodes:list,
                              edges:list,
                              costs:dict,
                              capacities:dict,
                              throughputs:dict,
                              demands:dict,
                              plants_n:list,
                              warehouses_n:list,
                              customers_n:list,
                              plants_to_warehouses:list,
                              plants_to_customers:list,
                              warehouses_to_customers:list):
    """Solves the minimum cost flow problem for the Crown Distributors Company example using Gurobi's ILP

    Args:
        nodes (list): List of nodes in the network. (plants, warehouses, customers)
        edges (list): List of edges in the network. (plants to warehouses, plants to customers, warehouse
        costs (dict): Dictionary of costs for each edge.
        capacities (dict): Dictionary of capacities for each plant.
        throughputs (dict): Dictionary of throughputs for each warehouse.
        demands (dict): Dictionary of demands for each customer.
        plants_n (list): List of plants.
        warehouses_n (list): List of warehouses.
        customers_n (list): List of customers.
        plants_to_warehouses (list): List of edges from plants to warehouses.
        plants_to_customers (list): List of edges from plants to customers.
        warehouses_to_customers (list): List of edges from warehouses to customers.

    Returns:
        None
    """
```

Λύση Πραγματικού Δικτύου

```
Objective value found: 198500.0
f[p1,w1] = -0.0
f[p1,w2] = 0.0
f[p1,w3] = -0.0
f[p1,w4] = 40000.0
f[p2,w2] = 50000.0
f[p2,w3] = 55000.0
f[p2,w4] = 0.0
f[p1,c1] = 50000.0
f[p1,c3] = -0.0
f[p1,c4] = -0.0
f[p1,c6] = 20000.0
f[p2,c1] = -0.0
f[w1,c2] = -0.0
f[w1,c3] = 0.0
f[w1,c4] = -0.0
f[w1,c6] = -0.0
f[w2,c1] = 0.0
f[w2,c2] = 10000.0
f[w2,c3] = -0.0
f[w2,c4] = 35000.0
f[w2,c5] = 5000.0
f[w3,c2] = -0.0
f[w3,c3] = -0.0
f[w3,c5] = 55000.0
f[w3,c6] = -0.0
f[w4,c3] = 40000.0
f[w4,c4] = -0.0
f[w4,c5] = 0.0
f[w4,c6] = -0.0
```

Ευχαριστώ για την Προσοχή σας