MIN COST FLOW PROBLEM

Μοντελοποίηση και Επίλυση

με Τεχνικές Γραμμικού Προγραμματισμού

ΓΕΩΡΓΙΟΣ ΠΑΠΟΥΤΣΑΣ

ΑΜ: 1083738

Έτος: 4ο

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών
Πανεπιστήμιο Πατρών
Ακαδημαϊκό Έτος 2023-2024

Περιεχόμενα

1. Εισαγωγή:

1.1 Επισκόπηση του Min Flow Cost Problem:

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.

1.2 Σημασία και Εφαρμογές του Προβλήματος:

Το πρόβλημα MCF αποτελεί γενίκευση πολλών γνωστών προβλημάτων βελτιστοποίησης, όπως το πρόβλημα της συντομότερης διαδρομής, το πρόβλημα της μέγιστης ροής και το πρόβλημα μεταφοράς. Βρίσκει εφαρμογές σε ένα ευρύ φάσμα πραγματικών σεναρίων, όπως στη διαχείριση εφοδιαστικής αλυσίδας και logistics, στις τηλεπικοινωνίες, στη δρομολόγηση κυκλοφορίας και στα χρηματοοικονομικά δίκτυα. Για παράδειγμα, στη διαχείριση logistics, το πρόβλημα μπορεί να αφορά στην εξεύρεση του πιο οικονομικού τρόπου διανομής αγαθών από πολλές αποθήκες σε διάφορες τοποθεσίες λιανικής, με στόχο την ελαχιστοποίηση του κόστους μεταφοράς και τη συμμόρφωση με τους περιορισμούς χωρητικότητας των οχημάτων.

2. Ορισμός του Προβλήματος:

Ένα δίκτυο ροής είναι ένας κατευθυνόμενος γράφος $G = (V, E)$ με ένα κόμβο source $s \in V$ και ένα κόμβο sink $t \in V$. Κάθε κόμβος $i \in V$ έχει προσφορά ή ζήτηση $s(i)$. Για $s(i) > 0$, λέμε ότι ο κόμβος έχει προσφορά και για $s(i) < 0$ έχει ζήτηση. Κάθε ακμή $(u, v) \in E$ έχει χωρητικότητα (capacity) $c(u, v) > 0$, ροή (flow) $f(u, v)$ και κόστος (cost) $a(u, v)$, με τους περισσότερους min cost flow αλγορίθμους να υποστηρίζουν ακμές με αρνητικά κόστη. Το κόστος μεταφοράς κάποιας ροής στην ακμή (u, v) είναι $f(u, v) \cdot a(u, v)$. Το πρόβλημα απαιτεί ένα ποσό ροής d να σταλθεί από το source s στο sink t.

Ο ορισμός του προβλήματος είναι να ελαχιστοποιήσει το συνολικό κόστος της ροής σε όλες της ακμές:

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

Υπό τους περιορισμούς:

Περιορισμοί χωρητικότητας: $f(u, v) \leq c(u, v)$

Αντισυμμετρικότητα: $f(u, v) = -f(v, u)$

Διατήρηση ροής: $\sum_{w\in V} f(u, w) = 0 \; για \; κάθε \; u \neq s, t$

Απαιτούμενη ροή: $\sum_{w\in V} f(s, w) = d \; και \; \sum_{w\in V} f(w, t) = d$

Το πρόβλημα δεν έχει λύση εάν:

$$\sum_{i\in V} s(i) \neq 0$$

3. Μοντελοποίηση με Γραμμικό Προγραμματισμό:

3.1 Διατύπωση του Min Flow Cost Problem ως Γραμμικό Πρόγραμμα:

Αντικειμενική συνάρτηση:

$$\min \sum_{e \in E} a(e) \cdot f(e)$$

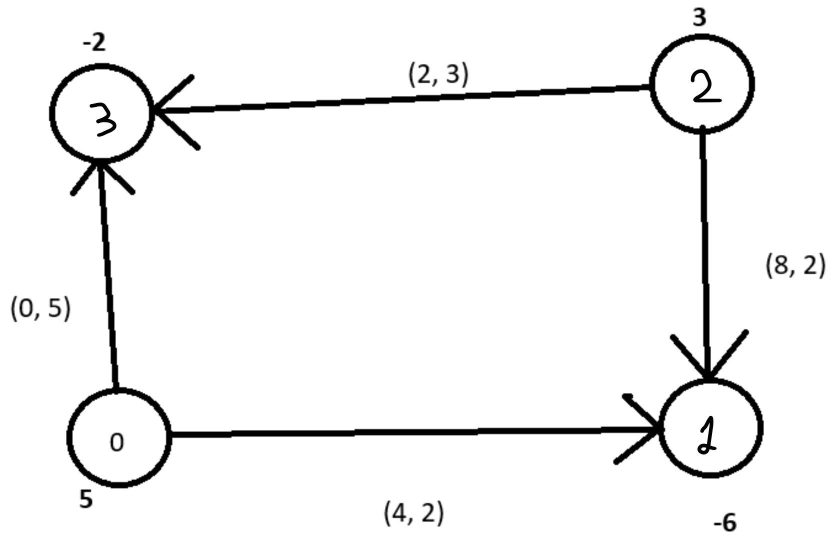
Περιορισμοί:

- Προσφορά και ζήτηση καλύπτονται για όλους τους κόμβους:
$$\sum_{j:(i,j)\in E} f(i,j) - \sum_{k:(k,i)\in E} f(k,i) = s(i) \; για \; κάθε \; i \in V$$

- Χωρητικότητες ακμών ικανοποιούνται:
$$f(i,j) \le c(i,j) \; για \; κάθε \; (i,j) \in E$$

- Οι ροές σε κάθε ακμή είναι μεγαλύτερες ή ίσες με 0:
$$f(i,j) \ge 0 \; για \; κάθε \; (i,j) \in E$$

3.3 Παράδειγμα Διατύπωσης ΓΠ για ένα Απλό Δίκτυο:



Εικόνα 1: Παράδειγμα δικτύου για διατύπωση του ΓΠ

Στην Εικόνα 1 φαίνεται ένα απλό δίκτυο ροής. Ο αριθμός με bold έξω από τον κόμβο είναι η προσφορά ή ζήτηση και σε κάθε ακμή υπάρχει ένα ζευγάρι (κόστος, χωρητικότητα).

Το πρόβλημα σύμφωνα με την ενότητα 3.1 διατυπώνεται ως εξής:

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

Υπό τους περιορισμούς:

Περιορισμοί διατήρησης ροής:

$$x_1 + x_2 = 5$$

$$x_1 + x_3 = 6$$

$$x_3 + x_4 = 3$$

$$-x_2 + x_4 = -2$$

Περιορισμοί χωρητικοτήτων:

$$0 \leq x_1 \leq 5$$

$$0 \leq x_2 \leq 2$$

$$0 \leq x_3 \leq 3$$

$$0 \leq x_2 \leq 2$$

Όπου:

x_1 είναι η ροή που θα σταλθεί από τον κόμβο $0 \rightarrow 1$

x_2 είναι η ροή που θα σταλθεί από τον κόμβο $0 \rightarrow 3$

x_3 είναι η ροή που θα σταλθεί από τον κόμβο $2 \rightarrow 1$

x_4 είναι η ροή που θα σταλθεί από τον κόμβο $2 \rightarrow 3$

4. Μέθοδοι Επίλυσης του Min Flow Cost Problem:

4.1 Αλγόριθμος Ακύρωσης Κύκλων:

Ο αλγόριθμος ακύρωσης κύκλων για την εύρεση ελάχιστου κόστους ροής σε ένα δίκτυο λειτουργεί εντοπίζοντας και ακυρώνοντας επανειλημμένα κύκλους αρνητικού κόστους στο υπολειπόμενο γράφημα, όπου κάθε κύκλος αντιπροσωπεύει μια διαδρομή μέσω της οποίας το συνολικό κόστος της ροής μπορεί να μειωθεί. Ξεκινώντας από μια αρχική εφικτή ροή, ο αλγόριθμος ανιχνεύει κύκλους αρνητικού κόστους (π.χ., με τον αλγόριθμο Bellman-Ford) και αυξάνει τη ροή κατά μήκος αυτών των κύκλων μέχρι να μην υπάρχουν άλλοι κύκλοι αρνητικού κόστους, εξασφαλίζοντας έτσι ότι η τελική ροή έχει το ελάχιστο δυνατό κόστος. Δεν σε αναλυθεί περαιτέρω, όμως στην ενότητα 7 υπάρχει ένα script που λύνει προβλήματα min cost flow με την βιβλιοθήκη or-tools της google που βασίζεται σε αυτόν τον αλγόριθμο.

4.2 Χρήση Ακέραιου Γραμμικού Προγραμματισμού και υλοποίηση σε Python:

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. Στο παράρτημα στην ενότητα 6 παρατίθεται ο κώδικας python που αναπτύχθηκε χρησιμοποιώντας την βιβλιοθήκη gurobipy.

5. Συμπεράσματα:

Συμπερασματικά, η μοντελοποίηση και επίλυση προβλημάτων ελάχιστου κόστους ροής μέσω γραμμικού προγραμματισμού προσφέρει μια ισχυρή και ευέλικτη προσέγγιση για τη βελτιστοποίηση της ροής σε δίκτυα. Χρησιμοποιώντας τις μεταβλητές απόφασης για να αναπαραστήσουμε τις ροές μεταξύ κόμβων, μπορούμε να ελαχιστοποιήσουμε το συνολικό κόστος ενώ σεβόμαστε τους περιορισμούς χωρητικότητας και ισοζυγίου προσφοράς-ζήτησης. Αυτή η μέθοδος επιτρέπει την εφαρμογή σε ποικίλα προβλήματα πραγματικού κόσμου, όπως οι μεταφορές, οι τηλεπικοινωνίες και η διαχείριση πόρων, προσφέροντας βέλτιστες λύσεις με αποτελεσματικό και μαθηματικά αυστηρό τρόπο.

6. Βιβλιογραφία

- https://en.wikipedia.org/wiki/Minimum-cost_flow_problem
- https://www.youtube.com/watch?v=r9L6CQRxgy0
- https://developers.google.com/optimization/flow/mincostflow#python
- https://www.topcoder.com/thrive/articles/Minimum%20Cost%20Flow%20Part%20Three:%20Applications

7. Παράρτημα Κώδικα:

Στο αρχείο integer_lp.py λύνεται το πρόβλημα με μεθόδους ακέραιου γραμμικού προγραμματισμού:

```python
import gurobipy as gp
from gurobipy import GRB
import time


def min_cost_flow_ilp(nodes, edges, cost, capacity, supply):
    start_time = time.time()

    # Suppress output
    env = gp.Env(empty=True)
    env.setParam("OutputFlag",0)
    env.start()

    model = gp.Model(env=env)

    ## Ορισμός μεταβλητών απόφασης
    f = model.addVars(nodes, nodes, vtype=GRB.INTEGER, name='f')

    ## Περιορισμοί για διατήρηση ροής
    for i in nodes:
        model.addConstr(gp.quicksum(f[i,j] for (x,j) in edges if x == i) -
                        gp.quicksum(f[j,i] for (j,x) in edges if x == i)
                        == supply[i])

    ## Περιορισμοί για τη ροή σε κάθε ακμή
    # Χωρητικότητα ακμών
    for (i,j) in edges:
        model.addConstr(f[i,j] <= capacity[i,j])

    # Θετικότητα ροής
    for (i,j) in edges:
        model.addConstr(f[i,j] >= 0)

    ## Αντικειμενική συνάρτηση
    model.setObjective(gp.quicksum(cost[i,j] * f[i,j] for (i,j) in edges), GRB.MINIMIZE)

    # Επίλυση
    model.optimize()

    end_time = time.time()

    diff = time.gmtime(end_time - start_time)
    print('\n[Total time used: {} minutes, {} seconds]'.format(diff.tm_min, diff.tm_sec))

```

```python
    # Εκτύπωση αποτελεσμάτων
    try:
        print(f'\nObjective value found: {model.objVal}')
    except AttributeError as e:
        print(f'\nCould not find an objective value. \nTraceback:\n\t{e}')


    # Βέλτιστη ροή ανά ακμή
    for (i,j) in edges:
        print('f[{},{}] = {}'.format(
            i, j, f[i,j].x
        ))



if __name__ == '__main__':
    # Input Data

    # Example_1
    '''
    MIN Z = x1 + 4x2 + 2x3 + 8x4
    subject to
    x1 + x2 = 5
    -x1 + -x3 = -6
    x3 + x4 = 3
    -x2 + x4 = -2
    x1 <= 5
    x2 <= 2
    x3 <= 3
    x4 <= 2
    and x1,x2,x3,x4 >= 0
    '''
    nodes1 = range(4)

    edges1 = [(0,1), (0,3), (2,1), (2,3)]

    # cost[i,j] is the cost of sending one unit of flow along edge (i,j)
    cost1 = {
    (0,1) : 0,
    (0,3) : 4,
    (2,1) : 2,
    (2,3) : 8
    }
```

```python
}

# capacity[i,j] is the capacity of edge (i,j)
capacity1 = {
(0,1) : 5,
(0,3) : 2,
(2,1) : 3,
(2,3) : 2
}

# supply[i] is the supply (if positive) or demand (if negative) of node i
supply1 = [5, -6, 3, -2]


# Example_2
nodes2 = range(5)

edges2 = [(0,1), (0,2), (1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (4,2)]

cost2 = {
    (0,1) : 4,
    (0,2) : 4,
    (1,2) : 2,
    (1,3) : 2,
    (1,4) : 6,
    (2,3) : 1,
    (2,4) : 3,
    (3,4) : 2,
    (4,2) : 3
}

capacity2 = {
    (0,1) : 15,
    (0,2) : 8,
    (1,2) : 20,
    (1,3) : 4,
    (1,4) : 10,
    (2,3) : 15,
    (2,4) : 4,
    (3,4) : 20,
    (4,2) : 5
}

supply2 = [20, 0, 0, -5, -15]

print("----------------------------Example_1----------------------------")
min_cost_flow_ilp(nodes1, edges1, cost1, capacity1, supply1)


print("\n----------------------------Example_2----------------------------")
min_cost_flow_ilp(nodes2, edges2, cost2, capacity2, supply2)
```

Στο αρχείο cycle_cancelling.py αξιοποιείται η βιβλιοθήκη or-tools της google:

```python
"""From Bradley, Hax and Maganti, 'Applied Mathematical Programming', figure 8.1."""
import numpy as np
from ortools.graph.python import min_cost_flow


def min_cost_flow_cc(start_nodes, end_nodes, capacities, unit_costs, supplies):
    """MinCostFlow simple interface example."""
    # Instantiate a SimpleMinCostFlow solver.
    smcf = min_cost_flow.SimpleMinCostFlow()

    # Add arcs, capacities and costs in bulk using numpy.
    all_arcs = smcf.add_arcs_with_capacity_and_unit_cost(
        start_nodes, end_nodes, capacities, unit_costs
    )

    # Add supply for each nodes.
    smcf.set_nodes_supplies(np.arange(0, len(supplies)), supplies)

    # Find the min cost flow.
    status = smcf.solve()

    if status != smcf.OPTIMAL:
        print("There was an issue with the min cost flow input.")
        print(f"Status: {status}")
        exit(1)
    print(f"Minimum cost: {smcf.optimal_cost()}")
    print("")
    print(" Arc    Flow / Capacity Cost")
    solution_flows = smcf.flows(all_arcs)
    costs = solution_flows * unit_costs
    for arc, flow, cost in zip(all_arcs, solution_flows, costs):
        print(
            f"{smcf.tail(arc):1} -> {smcf.head(arc)}  {flow:3}  / {smcf.capacity(arc):3}       {cost}"
        )


if __name__ == "__main__":


    # Example_1
    start_nodes1 = np.array([0, 0, 2, 2])
    end_nodes1 = np.array([1, 3, 1, 3])
    capacities1 = np.array([5, 2, 3, 2])
    unit_costs1 = np.array([1, 4, 2, 8])
    supplies1 = [5, -6, 3, -2]

    # Example_2
    start_nodes2 = np.array([0, 0, 1, 1, 1, 2, 2, 3, 4])
    end_nodes2 = np.array([1, 2, 2, 3, 4, 3, 4, 4, 2])
    capacities2 = np.array([15, 8, 20, 4, 10, 15, 4, 20, 5])
    unit_costs2 = np.array([4, 4, 2, 2, 6, 1, 3, 2, 3])
    supplies2 = [20, 0, 0, -5, -15]

    print("-----------------------------Example_1-----------------------------")
    min_cost_flow_cc(start_nodes1, end_nodes1, capacities1, unit_costs1, supplies1)

    print("-----------------------------Example_2-----------------------------")
    min_cost_flow_cc(start_nodes2, end_nodes2, capacities2, unit_costs2, supplies2)
```