Primjena LP-a za rješavanje problema maksimalnog protoka

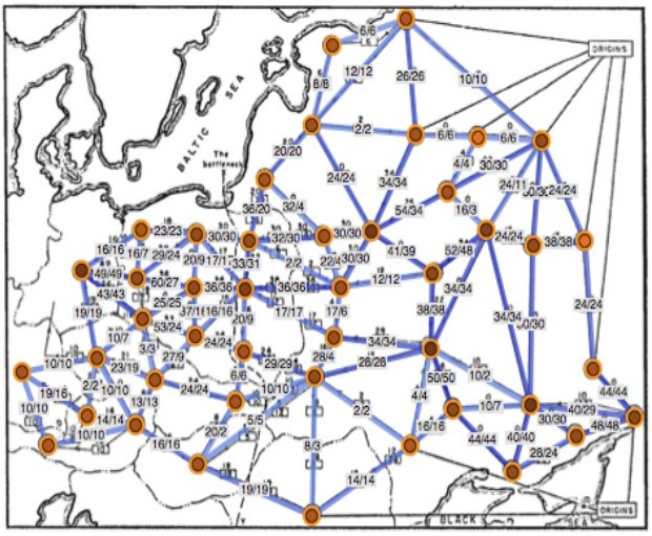
Emina Modrić

Ključne riječi: definicija problema maksimalnog protoka, grafički prikaz problema maksimalnog protoka, transportna mreža, ukupan protok, ograničenja kod problema maksimalnog protoka, linearno programiranje, python

I. Uvod

Mnogi problemi matematikog programiranja se mogu modelirati jezikom teorije grafova. Međutim, vrijedi i obratno, veliki broj problema teorije grafova se može prikazati kao modeli matematičkog programiranja, a često i kao modeli linearnog programiranja. Jedan od prvih problema tog tipa, koji je uspješno riješen metodima teorije grafova, a koji također spada i u specijalan slučaj problema linearnog programiranja je takozvani problem maksimalnog protoka.

Ranih 1950-tih, istraživači ratnog vazduhoplovstva SAD-a Ted Harris i Frank Ross objavili su izvještaj koji proučava željezničku mrežu između tadašnjeg Sovjetskog saveza i zemalja istočne Evrope. Ta željeznička mreža imala je 44 stanice i 105 pruga, pri čemu je svaka pruga imala kapacitet koji predstavlja maksimalnu količinu materijala koji se može poslati tom prugom iz jedne regije u drugu regiju u određenom vremenskom razdoblju. Struktura te mreže prikazana je na slici I.



Slika I. Problem maksimalnog protoka predstavljen željezničkom mrežom između Sovjetskog saveza i zemalja Istočne Evrope 1950ih [1]

Koristeći metodu pokušaja i pogreške, oni su odredili maksimalnu količinu materijala koji može da se prebaci iz Sovjeskog saveza u istočnu Evropu u posmatranom vremenskom razdoblju i najjednostavniji način da se prekinu veze u željezničkoj mreži, dizanjem u zrak minimalnog broja dijelova željezničke pruge. Skup grana (dijelova pruge) preko kojeg se zaustavlja prenos materijala, nazvali su usko grlo. Ovo je bila prva zabilježena aplikacija u praksi u kojoj se rješavao problem maksimalnog protoka.

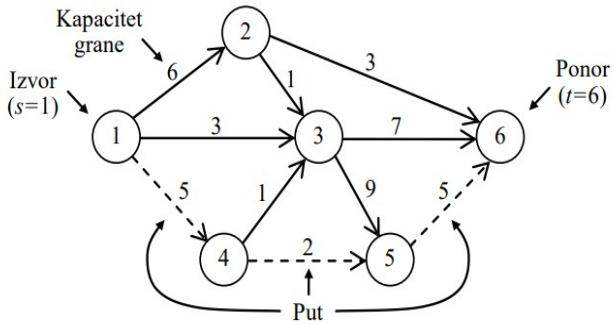
A. Opis problema

Problem maksimalnog protoka se na jednostavan način može predstaviti uz pomoć skupa cijevi koje imaju različite protočne kapacitete i spojene su na način da formiraju neku mrežu cijevi. Potrebno je odrediti kolika je maksimalna količina tečnosti koja može da se kroz takvu mrežu prenese od neke početne tačke (izvora) do krajnje tačke (odredišta). Jasno je da se takva mreža može modelirati kao usmjereni graf kod kojeg grane grafa predstavljaju cijevi protočne mreže, a čvorovi predstavljaju spojeve tih cijevi. Pri tome, svaka grana grafa ima određeni kapacitet kojim je definisana maksimalna količina koja može proći kroz nju (tj. kroz odgovarajuću cijev).

Za formalnu definiciju problema maksimalnog protoka potrebno je prvo uvesti pojam transportne mreže. Pod transportnom mrežom podrazumijeva se svaki usmjereni graf bez petlji kod kojeg postoji neki čvor s kojeg nazivamo izvor, u koji ne ulazi niti jedna grana grafa i neki čvor t kojeg nazivamo ponor, iz kojeg ne izlazi niti jedna grana grafa, i kod kojeg je svakoj grani (i, j) pridružen neki nenegativan broj $u_{i,j}$ nazvan kapacitet ili propusna sposobnost grane. Funkcija f(i, j) definisana na granama grafa (i, j) naziva se protok ili fluks ukoliko vrijedi $0 \leq f(i, j) \leq u_{i,j}$ i ukoliko je za svaki čvor k osim čvorova s i t, zbir vrijednosti f(i, k) za grane (i, k) koje ulaze u čvor k jednak zbiru vrijednosti f (k, j) za grane (k, j) koje izlaze iz čvora k, odnosno:

$$\sum_{(i,k)\in R} f(i,k) = \sum_{(k,j)\in R} f(k,j)\,,\ k \in \{1,2,\dots,n\} \setminus \{s,t\}$$

Ova relacija predstavlja zakon o konzervaciji protoka i odgovara prvom Kirchoffovom zakonu u teoriji električnih kola. Ukupan protok kroz transportnu mrežu definiše se kao zbir protoka grana koje izlaze iz čvora s i zbira protoka grana koje ulaze u čvor t (lako se pokazuje da ovo dvoje mora biti jednako). Na slici II prikazan je primjer grafa.



Slika II. Grafički prikaz transportne mreže [2]

B. *Pregled literature vezane za opisani problem*

Za rješavanje ovog problema u najvećoj mjeri korištena su predavanja sa predmeta Osnove operacionih istraživanja od profesora doktora Željka Jurića. Među njima, najviše su korištena predavanja Linearno programiranje [3] te predavanje Teorija grafova i linearno programiranje. U njima je pronađen iscrpan uvod u problematiku koji je bio dovoljan da se adekvatno pristupi rješavanju zadatka. Također, korištena su i predavanja sa predmeta Optimalno upravljanje od profesora doktora Samima Konjicije[4], a posebno materijali za linearno programiranje. Pored ovoga, korištena su i predavanja o formulaciji problema linearnog programiranja iz problema maksimalnog protoka sa Univerziteta u Stanfordu [5].

C. *Moguće aplikacije u praksi*

Zadatak nalaženja maksimalnog protoka se najčešće susreće u raznim tipovima realnih transportnih zadataka. Na primjer, ako je iz jednog grada (početna tačka grafa) za određeno vrijeme potrebno prebaciti što više putnika u drugi grad (krajnja tačka grafa) mrežom avionskog saobraćaja, dobija se upravo zadatak maksimalnog protoka. Podrazumijeva se da svaka relacija avionskog saobraćaja ima svoju propusnu sposobnost, tj. gornju granicu tereta koji se može po njoj prevesti za određeno vrijeme. Pri tome se pretpostavlja da se putnici ne mogu ni ukrcavati ni iskrcavati u međustanicama, inače se zadatak usložnjava. Pored ovoga, postoje mnogobrojni primjeri šta u realnim primjenama mogu biti mreže, čvorovi grane i protoci. Naprimjer, u komunikacionim mrežama čvorove mogu predstavljaju računari i saletili, grane kablovi i optička vlakna a protoci mogu biti audio ili video paketi. Ukoliko se mrežom predstavljaju električna kola, čvorovi mogu biti registri i sabirnice, grane žice a protok struja. Primjer mreže se može naći i u hidraulici gdje rezervoari i pumpe predstavljaju čvorove, cijevi grane a fluidi protoke. Velike primjene problema maksimalnog protoka mogu se naći i u finansijama gdje su čvorovi dionice i valute, transakcije su predstavljene granama a novac predstavlja protok, a zatim i u transportu gdje su čvorovi predstavljeni aerodromima i stanicama, grane su autoputevi i pruge a protok su teret ili putnici.

II. KORIŠTENI ALGORITAM

Kao što smo vidjeli, problem maksimalnog protoka je jedan od najvažnijih problema optimizacije nad grafovima. Zbog aktuelnosti problema, ovaj problem je predmet intenzivnih proučavanja i stalno se razvijaju sve bolji algoritmi za njihovo rješavanje. Neki od najvažnijih pobrojani su u tabeli I, pri čemu rubrika "Red složenosti" označava red veličine broja operacija u najgorem slučaju, pri čemu se često rješenje nalazi znatno brže (oznaka C u tabeli označava kapacitet mreže, osnosno sumu kapaciteta svih grana koje izlaze iz izvora).

TABELA I. HISTORIJSKI RAZVOJ ALGORITAMA ZA RJEŠENJE PROBLEMA MAKSIMALNOG PROTOKA

Godina	Otkrio	Metod	Red složenosti
1951	Dantig	Simplex	mnC
1955	Ford, Fulkerson	Augmenting Path	mF_{max}
1970	Dinitz	Blocking Path	nm^2
1972	Edmonds, Karp	Shortest Path	nm^2
1972	Edmonds, Karp, Dinitz	Capacity Scaling	$m^2\log_2 C$
1973	Dinitz, Gabow	Capacity Scaling	$mn\log_2 C$
...	...	...	...
2012	Orlin	Compact Abudance Graphs	mn

A. *Opis rada korištenog algoritma*

Sada će se preći na predstavljanje zadatka maksimalnog protoka u obliku zadatka linarnog programiranja. Neka je n broj čvorova grafa. Bez umanjenja općenitosti može se uzeti da je izvor (početni čvor) označen sa 1, a ponor (krajnji čvor) sa n. Neka promjenljive $x_{i,j}$ predstavljaju protoke kroz grane koje vode od čvora i do čvora j i neka su $u_{i,j}$ kapaciteti odgovarajućih grana (ukoliko čvor i i j nisu spojeni granom, može se uzeti da je kapacitet te grane nulti, odnosno $u_{i,j} = 0$). Ukoliko sa F označimo ukupni protok kroz mrežu, tada se zadatak nalaska maksimalnog protoka može zapisati u obliku

$$\arg max\ F = \sum_{j=2}^{n} x_{1,j}$$

$$p.o.\ \sum_{i=1}^{n} x_{i,k} - \sum_{j=1}^{n} x_{k,j} = 0, k = 2,..,n-1$$

$$0 \le x_{i,j} \le u_{i,j}, i = 1,..,n, \qquad j = 1,..,n$$

Prva skupina ograničenja predstavljaju jednačine očuvanja toka za sve ostale čvorove osim izvora i ponora. One odražavaju činjenicu da u tim čvorovima ne nastaje niti nestaje tok, tako da ukupan tok koji ulazi u čvor mora biti jednak ukupnom toku koji izlazi iz čvora. Druga skupina ograničenja predstavljaju ograničenja maksimalnog toka kroz grane u skladu sa njihovim kapacitetima. Nekada se prvoj skupini ograničenja dodaje i ograničenje

$$\sum_{j=2}^{n} x_{1,j} - \sum_{i=1}^{n-1} x_{i,n} = 0$$

koje izražava činjenicu da je ukupan protok koji prolazi kroz mrežu jednak bilo sumi protoka koji izlaze iz početne tačke grafa, bilo sumi protoka koji ulaze u krajnju tačku grafa. Međutim, ovo ograničenje je suvišno (odnosno redudantno), jer se ono može izvesti kao posljedica ostalih ograničenja iz ove skupine.

Može se još dodati da problem maksimalnog protoka, slično transportnom problemu, ima matricu ograničenja koja je totalno unimodularna (ona se efektivno sastoji od dvije submatrice od kojih je jedna matrica incidencije polaznog grafa, dok je druga jedinična matrica). Slijedi da problemi maksimalnog protoka kod kojih su kapaciteti svih grana cjelobrojni, uvijek imaju cjelobrojno optimalno rješenje. Drugim riječima, eventualna dodatna ograničenja na cjelobrojnost protoka kroz pojedine grane mreže ni na kakav način ne utiču na rješavanje problema.

B. Svođenje opisanog problema u formu korištenog algoritma

U ovom dijelu teksta bit će predstavljen i pojašnjen kod koji izvršava ovaj algoritam.

Transportna mreža između čvorova koji ju sačinjavaju, a koja se odvija preko grana između njih, predstavljena je matricom incidencije.

Matricu incidencije je potrebno transformisati tako da se kreiraju funkcija cilja i ograničenja. Matrica incidencije je kvadratna matrica dimenzija nxn. Ovakva transportna mreža može biti modelirana preko n^2 grana, pri čemu je po jedna grana dodijeljena za svaka dva čvora u grafu. Za rješavanje problema linearnog programiranja potrebno je kreirati matricu A i vektor b koji predstavljaju ograničenja tipa nejednakosti, pri čemu vrijedi:

$A \cdot x \leq b$

Ova ograničenja modeliraju ograničenja maksimalnog toka kroz grane u skladu sa njihovim kapacitetima zadanim matricom incidencije. Stoga matrica A predstavlja jediničnu matricu dimenzija $n^2 x n^2$, a vektor b ima dimenzije $1 x n^2$ i sastavljen je od elemenata matrice incidencije. Zatim, potrebo je kreirati matricu A_{eq} i vektor b_{eq} koji predstavljaju ograničenja tipa jednakosti, pri čemu vrijedi:

$A_{eq} \cdot x \leq b_{eq}$

Ova ograničenja predstavljaju jednačine očuvanja toka za sve ostale čvorove osim izvora i ponora, pa stoga matrica A_{eq} ima dimenzije $(n-2) x n^2$, a vektor b_{eq} je dimenzije $1 x (n-2)$. I-ti red matrice A_{eq} je kreiran uz pomoć i-te kolone i i-tog reda matrice incidencije, naime, nenulti elementi i-te kolone matrice incidencije će u matrici A_{eq} biti predstavljeni vrijednošću 1, a nenulti elementi i-tog reda matrice incidencije će u matrici A_{eq} biti predstavljeni vrijednošću -1. Vektor b_{eq} ima sve elemente nulte.

Vektor c, koji predstavlja funkciju cilja, će također biti dimanzije $1 x n^2$, pri čemu će nenulti elementi prvog reda matrice incidencije imati vrijednost -1, a svi ostali elementi će imati vrijednost 0. (Vrši se maksimizacija problema a funkcija linprog vrši minimizaciju, stoga su svi elementi funkcije cilja pomnoženi sa -1). Kod ove funkcije napisan u programskom jeziku Python dat je ispod.

```
def transformisi_matricu(M):
    n = len(M)
    A = np.identity(n**2)
    b = [0] * (n**2)
    Aeq = np.zeros((n, n**2))
    beq = np.zeros((n-2,1))
    C = np.zeros((1,n**2))

    for i in range(n):
        for j in range(n):
            b[i*n + j] = M[i][j]

            if M[i][j] != 0:
                Aeq[i][i*n + j] = -1
                Aeq[j][i*n + j] = 1

    for i in range(n):
        if M[0][i] != 0:
            C[0][i] = -1

    Aeq = np.delete(Aeq, (0, n-1), axis=0)
    return (C,A,b,Aeq,beq)
```

Da bi se kod mogao izvršiti potrebno je instalirati numpy pokretanjem komande pip install numpy u komandnom prozoru. Numpy je u prethodni kod importovan naredbom

```
import numpy as np
```

Sada kada su kreirane matrice i vektori koji transformišu ovaj problem u problem linearnog programiranja, potrebno je pozvati funkciju linprog. Funkcija linprog rješava probleme linearnog programiranja minimizirajući ih, a njeni parametri su upravo ove matrice i vektori koji su iznad kreirani. Za pokretanje funkcije linprog potrebno je instalirati dodatak scipy pokretanjem naredbe pip install scipy u komandnom prozoru. Narednom linijom koda funkcija linprog je importovana da bi se mogla koristiti u kodu.

```
from scipy.optimize import linprog
```

Poziv funkcije za transformisanje matrice a nakon toga i funkcije linprog prikazan je narednim isječkom.

```
(C,A,b,Aeq,beq) = transformisi_matricu(M1)
res = linprog(C, A_ub=A, b_ub = b, A_eq = Aeq, b_eq =
beq)
```

Funkcija linprog će vratiti vektor dimenzija $1 x n^2$. Potrebno je ovaj vektor transformisati nazad u matricu koja će prikazati koliki je protok kroz odgovarajuću granu. Za ovo je napisana funkcija transformisi_u_matricu čiji je kod dat ispod.

```
def transformisi_u_matricu(A,n):
    M = np.zeros((n,n))
    for i in range(n):
        M[i] = A[(n*i):(n*i+n)]
    return M
```

Još je dodana i funkcija za zaokruživanje rezultata na dvije decimale kako bi rezultati bili pregledniji. Konačno, nakon poziva funkcije linprog potrebno je rezultate transforisati u željeni oblik što se postiže narednim isječkom koda.

```
X = zaokruzi_vrijednosti(transformisi_u_matricu(res.x,
len(M1)))
Z = zaokruzi_vrijednosti(-1 * res.fun)
```

Ovime je u potpunosti objašnjeno svođenje opisanog problema u formu korištenog algoritma.

III. SIMULACIJSKI REZULTATI

Rad prethodno predstavljenog algoritma bit će prikazan kroz tri primjera, koji zapravo predstavljaju tipične praktične primjere upotrebe problema maksimalnog protoka, a to su komunikacione mreže, finansije i transport.

A. Postavka simulacija

Prvi problem koji će biti prikazan jeste problem maksimalnog protoka kroz komunikacionu mrežu. Neka je potrebno simulirati komunikacionu mrežu između osam računara. Na prvom računaru, koji je sa ostalima umrežen kablovima, se kreiraju video paketi. Video paketi se prosljeđuju komunikacionom mrežom do osmog, odnosno posljednjeg računara, koji predstavlja ponor. Potrebno je postići maksimalan protok kroz ovu mrežu. Matrica incidencije data je u tabeli II.

TABELA II. MATRICA INCIDENCIJE

0	3	2	2	0	0	0	0
0	0	0	0	5	1	0	0
0	0	0	0	1	3	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0

U main.py dio programa je potrebno unijeti ovu matricu incidencije. Njen unos i poziv dati su kodom ispod:

```
M1= [[0,3,2,2,0,0,0,0],
    [0,0,0,0,5,1,0,0],
    [0,0,0,0,1,3,1,0],
    [0,0,0,0,0,1,0,0],
    [0,0,0,0,0,0,0,4],
    [0,0,0,0,0,0,0,2],
    [0,0,0,0,0,0,0,4],
    [0,0,0,0,0,0,0,0]]

print("Simulacija problema maksimalnog protoka kroz
komunikacionu mrežu")
(C,A,b,Aeq,beq) = transformisi_matricu(M1)
res = linprog(C, A_ub=A, b_ub = b, A_eq = Aeq, b_eq =
beq)

X = zaokruzi_vrijednosti(transformisi_u_matricu(res.x,
len(M1)))
Z = zaokruzi_vrijednosti(-1 * res.fun)

print(X)
print(Z)
```

Naredni problem koji će biti posmatran je problem maksimalnog protoka u finansijama. Neka je potrebno simulirati transakcijsku mrežu između šest dionica među kojima se obavljaju transakcije, među kojima je prva dionica izvor a posljednja, odnosno šesta, ponor. Maksimalne količine novca izražene u hiljadama koje mogu biti prenesene između određenih dionica date su matricom incidencije koja je prikazana u tabeli III.

TABELA II. MATRICA INCIDENCIJE

0	20	60	0	0	0
0	0	50	30	10	0
0	0	0	35	0	10
0	0	0	0	30	25
0	0	0	0	0	50
0	0	0	0	0	0

U main.py dio programa je potrebno unijeti ovu matricu incidencije. Njen unos i poziv dati su kodom ispod:

```
M2= [[0,20,60,0,0,0],
    [0,0,50,30,10,0],
    [0,0,0,35,0,10],
    [0,0,0,0,30,25],
    [0,0,0,0,0,50],
    [0,0,0,0,0,0]]

print("Simulacija problema maksimalnog protoka u
finansijama")
(C,A,b,Aeq,beq) = transformisi_matricu(M2)
res = linprog(C, A_ub=A, b_ub = b, A_eq = Aeq, b_eq =
beq)

X = zaokruzi_vrijednosti(transformisi_u_matricu(res.x,
len(M2)))
Z = zaokruzi_vrijednosti(-1 * res.fun)

print(X)
print(Z)
```

Posljednji problem koji će biti posmatran u sklopu ovog rada je transportni problem maksimalnog protoka. Neka je potrebno simulirati transportni problem između sedam gradova koji su povezani autoputem. Potrebno je kamionima prenijeti maksimalan teret od snabdjevača u prvom gradu koji predstavlja izvor do magacina koji se nalazi u posljednjem gradu i predstavlja ponor. Maksimalni teret izražen u tonama koji može biti transportovan između svaka dva odgovarajuća

grada dat je matricom incidencije koja je prikazana tabelom IV.

TABELA IV. MATRICA INCIDENCIJE

0	3	1	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	3.5	0
0	0	0	0	4	4	0
0	0	0	0	0	0	4
0	0	0	2.5	0	0	1
0	0	0	0	0	0	0

U main.py dio programa je potrebno unijeti ovu matricu incidencije. Njen unos i poziv dati su kodom ispod:

```
M3= [[0,3,1,1,0,0,0],
    [0,0,1,0,1,0,0],
    [0,1,0,0,0,3.5,0],
    [0,0,0,0,4,4,0],
    [0,0,0,0,0,0,4],
    [0,0,0,2.5,0,0,1],
    [0,0,0,0,0,0,0]]
```

B. *Rezultati simulacija*

Nakon pokretanja dijela koda koji odgovara prvom zadatku na ekranu se dobije rezultujući ispis koji je prikazan na slici III ispod.

```
Simulacija problema maksimalnog protoka kroz komunikacionu mrežu
[[0.   3.   2.   1.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   2.74 0.26 0.   0.  ]
 [0.   0.   0.   0.   0.71 0.53 0.75 0.  ]
 [0.   0.   0.   0.   0.   1.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   3.45]
 [0.   0.   0.   0.   0.   0.   0.   1.79]
 [0.   0.   0.   0.   0.   0.   0.   0.75]
 [0.   0.   0.   0.   0.   0.   0.   0.  ]]
6.0
```

Slika III. Rezultat simulacije problema maksimalnog protoka kroz komunikacionu mrežu

Ukupni tok od izvora do ponora je 6, dok su količine toka kroz odgovarajuću granu čitljive sa slike. Ovo rješenje je ujedno i optimalno rješenje pri kojem se postiže maksimalan protok od izvora do ponora.

Nakon pokretanja dijela koda koji odgovara drugom zadatku na ekranu se dobije rezultujući ispis koji je prikazan na slici IV ispod.

```
Simulacija problema maksimalnog protoka u finansijama
[[ 0.   20.   45.    0.    0.    0.  ]
 [ 0.    0.    0.   13.16  6.84  0.  ]
 [ 0.    0.    0.   35.    0.   10.  ]
 [ 0.    0.    0.    0.   26.35 21.81]
 [ 0.    0.    0.    0.    0.   33.19]
 [ 0.    0.    0.    0.    0.    0.  ]]
65.0
```

Slika IV. Rezultat simulacije maksimalnog protoka u finansijama

Ukupni tok kroz ovaj graf iznosi 65 hiljada novca neke valute. Veličine transakcije između svake dvije dionice su čitljive iz matrice sa prikazane slike. Ovo rješenje je optimalno rješenje problema maksimalnog protoka primijenjenog u finansijama.

Nakon pokretanja dijela koda koji odgovara trećem zadatku na ekranu se dobije rezultujući ispis koji je prikazan na slici V ispod.

```
Simulacija transportnog problema maksimalnog protoka
[[0.   2.   1.   1.   0.   0.   0.  ]
 [0.   0.   1.   0.   1.   0.   0.  ]
 [0.   0.   0.   0.   0.   2.   0.  ]
 [0.   0.   0.   0.   2.33 0.62 0.  ]
 [0.   0.   0.   0.   0.   0.   3.33]
 [0.   0.   0.   1.95 0.   0.   0.67]
 [0.   0.   0.   0.   0.   0.   0.  ]]
4.0
```

Slika V. Rezultat simulacije transportnog problema maksimalnog protoka

Postignuto je optimalno rješenje tako da se nijednom dionicom autoputa ne prenosi veći teret od dozvoljenog. Ukupni teret koji se može prenijeti od snabdjevača do magacina iznosi 4 tone, a količina robe u tonama koja se prevozi određenom dionicom autoputa je čitljiva sa slike.

C. *Zaključak*

U prikazanim simulacijama zaista je postignuto optimalno rješenje. Ovo je provjereno koristeći razne solvere za rješavanje problema maksimalnog protoka. Prikazano korištenje funkcije je veoma jednostavno. U suštini, za zadavanje problema potrebno je samo zadati matricu incidencije transportnog problema. Algoritam uspješno ovu matricu transformiše u funkciju cilja i ograničenja linearnog programiranja te se poziva funkcija linprog koja adekvatno rješava problem linearnog programiranja.

IV. ZAKLJUČAK I DISKUSIJA

U ovom radu prikazana je formulacija problema maksimalnog protoka u obliku linearnog programiranja. Za rješavanje problema linearnog programiranja moguće je koristiti Simpleks algoritam ili neki drugi algoritam.

Linearno programiranje je veoma učinkovita metoda za rješavanje problema maksimalnog protoka. Ovaj algoritam uvijek dolazi do optimalnog rješenja problema maksimalnog protoka. Metod linearnog programiranja je veoma izučen, što predstavlja veliki plus za rješavanje problema ovim metodom.

Za rješavanje ovim metodom nisu potrebne memorijske zalihe. Međutim, red složenosti ovog algoritma je mnC, gdje je m broj čvorova, n broj grana a C kapacitet mreže, odnosno suma kapaciteta svih grana koje izlaze iz izvora.

Problem linearnog programiranja koji je izveden iz problema maksimalnog protoka ima veoma velik broj

promjenljivih (n^2 promjenljivih, pri čemu grana između svaka dva čvora predstavlja jednu promjenljivu). Moguća je optimizacija ovog algoritma da se izbaci višak promjenljivih.

Druga stvar je da problemi maksimalnog protoka sa većim dimenzionalnostima predstavljeni kao problemi linearnog programiranja imaju prevelik broj ograničenja. Stoga je razvijen poseban algoritam linearnog programiranja za rješavanje problema maksimalnog protoka koji se naziva „Network“ Simplex Method.

Danas postoje i znatno efikasniji ali i znatno složeniji algoritmi za nalaženje maksimalnog protoka, u odnosu na algoritam prikazan u ovom radu. Relativno je jednostavni su još i Ford-Fulkerson, Edmonds-Karp, te Dinitzov algoritam. Poznate su i razne verzije algoritama zasnovane na korištenju tzv. predprotoka (engl. preflow) koji predstavljaju funkcije nad granama slične protocima, ali koje ne zadovoljavaju zakon o konzervaciji toka, nego dopuštaju “viškove” u čvorovima grafa. Prvi algoritam tog tipa predložio je A. V. Karzanov, a složenost izvedbe takvih algoritama varira od relativno jednostavne do izuzetno komplikovane. Do sada je po pitanju efikasnosti najviše odmakao J. Orlin čiji vrlo komplikovani algoritam zasnovan na iznimno komplikovanim strukturama podataka rješava problem maksimalnog protoka u vremenu koje ne prelazi iznos proporcionalan sa mn.

REFERENCE

[1] J. Y. Yu, “Network Flow Lectures, Concordia University, November 19, 2015 pp. 1–4.

[2] T. Mateljan, Ž. Jurić, Osnove operacionih istraživanja, Radni materijali za kurs “Osnove operacionih istraživanja” na Elektrotehničkom fakultetu u Sarajevu, Predavanje Teorija grafova i linearno programiranje, Akademska godina 2014/15, pp. 1-29.

[3] T. Mateljan, Ž. Jurić, Osnove operacionih istraživanja, Radni materijali za kurs “Osnove operacionih istraživanja” na Elektrotehničkom fakultetu u Sarajevu, Predavanje Linearno programiranje, Akademska godina 2014/15, pp. 1-59.

[4] S. Konjicija, Radni materijali sa predavanja za kurs „Optimalno upravljanje“, materijali o linearnom programiranju, 01. April 2019, pp. 1-29

[5] L. Trevisan, „Optimisation – Lecture 15“, Stanford University, February 24, 2011, pp. 1-7