PROJEKT PRI FINANČNEM PRAKTIKUMU

Tema 3

Avtorici: Marina Sirk in Nataša Taškov

ORIGINALNO NAVODILO

Consider a set P of 2n points in the plane. A shortest matching for P is a set of segments with minimum length such that each point of P is an endpoint of precisely one segment. Formulate the problem as an ILP. Make experiments to find the expected value of total length of the shortest matching when the points are selected uniformly at random on the unit square, unit disk, etc. Does the value increase or decrease when n grows? Consider the problem when the set of points consists of n red and n blue points and the matching has two use segments whose endpoints have different colors (bichromatic shortest matching). Compare the value of the monochromatic to the bichromatic solution.

OPIS PROBLEMA

Za množico P z 2n točkami v ravnini je prirejanje na P definirano kot množica n daljic s krajišči v točkah iz P, pri čemer je vsaka točka iz P krajišče natanko ene daljice (tj., nobeni dve daljici nimata skupnega krajišča). Zanima nas najkrajše prirejanje na P, torej tako prirejanje na P, pri katerem je vsota dolžin njegovih daljic najkrajša.

Pri prvem problemu (»monochromatic shortest matching«) imamo ravnino z 2n točkami enake barve. Povezati moramo po dve in dve točki skupaj oziroma izbrati n daljic tako, da bo skupna dolžina daljic najkrajša in bo hkrati vsaka točka krajišče natanko ene daljice.

Naslednji problem (»bichromatic shortest matching«) pa ima ravno tako 2n točk v ravnini, vendar je od teh n rdečih in n modrih. Ravno tako moramo izbrati n daljic, pri čemer ima daljica za krajišči eno modro in eno rdečo točko.

NAČRT ZA REŠEVANJE PROBLEMA 1

Minimizirati želimo skupno dolžino daljic, zato bo pri našem ILP veljalo:

$$min \sum_{i=1}^{2n} \sum_{j=i+1}^{2n} x_{i,j} a_{i,j},$$

pri čemer so $a_{i,j}$ elementi matrike

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,2n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{2n,1} & a_{2n,2} & \cdots & a_{2n,2n} \end{bmatrix},$$

kjer nam $a_{i,j}$ pove razdaljo med točko i in j. $x_{i,j}$ pa so elementi matrike

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,2n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2n,1} & x_{2n,2} & \cdots & x_{2n,2n} \end{bmatrix}, x_{i,j} \in \{0,1\}$$

in $x_{i,j}$ zasede vrednost 1, če sta točki i in j povezani, sicer pa vrednost 0.

Pri tem linearnem programu moramo predpostaviti še nekaj pogojev, in sicer:

$$\sum_{j}^{2n} x_{i,j} = 1 \forall i \in \{1,2,\dots,2n\}.$$

Ta pogoj poskrbi, da je vsaka točka povezana z natanko eno drugo točko. Veljati pa mora še:

$$x_{i,j} \in \{0,1\}; i,j \in \{1,2,\dots,2n\},$$

ta pogoj pove, da so elementi $x_{i,j}$ lahko samo 0 ali 1,

$$x_{i,j} = x_{j,i}; i,j \in \{1,2,\dots,2n\},$$

ta pogoj pomeni, da je matrika X simetrična in še zadnji pogoj

$$x_{i,i} = 0; i \in \{1,2,\dots,2n\}$$

poskrbi, da so diagonalni elementi enaki 0.

NAČRT ZA REŠEVANJE PROBLEMA 2

Zdaj imamo n rdečih in n modrih kroglic v ravnini. Problem je podoben problemu 1, le da morajo biti vse daljice sestavljene iz ene rdeče in ene modre kroglice. Ponovno sestavimo matriko razdalj A, pri čemer prvih n vrstic in stolpcev predstavlja rdeče kroglice, drugih n pa modre.

Problema se lotimo z enakim algoritmom kot pri problemu 1, le da ga uporabimo na podmatrikah matrike A, ki zadoščajo pogojem:

$$a_{i,j}; i \in \{1,2,\dots,n\}, j \in \{n+1, n+2,\dots,2n\}$$

in

$$a_{i,j}; i \in \{n+1, n+2,\dots,2n\}, j \in \{1,2,\dots,n\}$$

Na ta način zagotovimo, da gledamo le povezave med eno rdečo in eno modro kroglico naenkrat.

IZBIRA DELOVNEGA OKOLJA

Za reševanje najinega problema sva uporabile programsko okolje Sage oziroma CoCalc, ki je dostopno na povezavi https://cocalc.com/. Za to storitev sva se odločili, ker je v njem lažje reševati probleme celoštevilskega linearnega programiranja in omogoča urejevanje datotek več osebam hkrati z različnih računalnikov, kar pomeni lažje sodelovanje.

POTEK REŠEVANJA

Najprej sva definirali funkcijo, ki generira 2n naključnih točk v ravnini, pri tem je n poljuben in ga je potrebno izbrati. Poleg tega pa si izberemo še območje, na katerem želimo, da ležijo naše točke, torej izberemo interval (a, b).

```
def tocke(n, a, b):
    points = []
    i = 1

    while i <= 2*n:
        (x, y) = (random.uniform(a, b), random.uniform(a, b))
        points.append((x, y))
        i += 1

    return points
```

Nato sva definirali še funkcijo, ki izpiše matriko povezav med točkami.

```
def matrika_povezav(points):
    A = []
    k = 0
    stevilo_tock = len(points)

    while k <= (stevilo_tock - 1):
        point = points[k]       # k-ta točka
        x_koord = point[0]      # x koordinata k-te točke
        y_koord = point[1]      # y koordinata k-te točke
        razdalje = []           # razdalje od k-te točke do vseh točk v ravnini

        for tocka in points:
            x = tocka[0]
            y = tocka[1]
            dolzina = math.sqrt((x_koord - x)**2 + (y_koord - y)**2)
            razdalje.append(dolzina)

        A.append(razdalje)
        k += 1

    return A
```

Potem sva napisali kodo za linearni program. S črko t je označen seznam točk v ravnini, z A pa je označena matrika povezav.

```
t = tocke(n, a, b)
A = matrika_povezav(t)
d = len(A)
```

Koda za problem 1 je sledeča:

```
p = MixedIntegerLinearProgram(maximization = False)
X = p.new_variable(binary = True)

p.set_objective(sum(sum(A[i][j]*X[i, j] for j in range(i, d)) for i in range(d)))

# pogoj 1
for i in range(d):
    p.add_constraint(sum(X[i, j] for j in range(d)) == 1)

# pogoj 2
for i in range(d):
    for j in range(d):
        p.add_constraint(X[i, j] == X[j, i])

# pogoj 3
for i in range(d):
    p.add_constraint(X[i, i] == 0)

min_dolzina1 = p.solve()

resitev1 = p.get_values(X)

Resitev1 = []
for i in range(d):
    for j in range(d):
        if resitev1[i, j] == 1:
            Resitev1.append((i + 1,j + 1))
```

Na začetku povemo, da gre za minimizacijo in da je matrika X sestavljena iz ničel in enic. Sledi dodajanje pogojev; pogoj 1 poskrbi, da vsako točko povežemo z natanko eno točko, pogoj 2 naredi matriko X simetrično, pogoj 3 pa dodamo, ker točke ne dovolimo povezati same s seboj. Nato dobimo rešitev z ukazom *p.solve()*, to je najmanjša skupna dolžina povezav med točkami, torej vsota vseh dolžin daljic. Če želimo izvedeti še kako so točke povezane med seboj, torej katera točka je s katero povezana, to izvemo z ukazom *p.get.values(),* še lepše pa nam to pove *Resitev1*.

Za problem 2 je koda podobna, le da lahko nekaj pogojev izpustimo, saj sedaj namesto matrike A, uporabimo matriko B, ki je zgornja desna četrtina matrike A. Tako res upoštevamo samo povezave med rdečimi in modrimi točkami. Paziti moramo, ker se spremeni velikost matrike iz *d* na *d/2*. Dodati moramo samo še pogoja, da mora biti vsota po vseh vrsticah enaka 1 in prav tako mora biti tudi vsota po vseh stolpcih enaka 1, kar pomeni, da je vsaka točka povezana z natanko eno drugo točko.

```
A = numpy.array(A)
B = A[:d/2, d/2:]     # vrstice do izključno d/2, stolpci od vključno d/2

q = MixedIntegerLinearProgram(maximization = False)
Y = q.new_variable(binary = True)

q.set_objective(sum(sum(B[i][j]*Y[i, j] for j in range(d/2)) for i in range(d/2)))

# pogoja
for i in range(d/2):
    q.add_constraint(sum(Y[i, j] for j in range(d/2)) == 1)        # vsota po stolpcih enaka 0
for j in range(d/2):
    q.add_constraint(sum(Y[i, j] for i in range(d/2)) == 1)        # vsota po vrsticah enaka 0

min_dolzina2 = q.solve()

resitev2 = q.get_values(Y)

Resitev2 = []
for i in range(d/2):
    for j in range(d/2):
        if resitev2[i, j] == 1:
            Resitev2.append((i + 1, d/2 + j + 1))
```

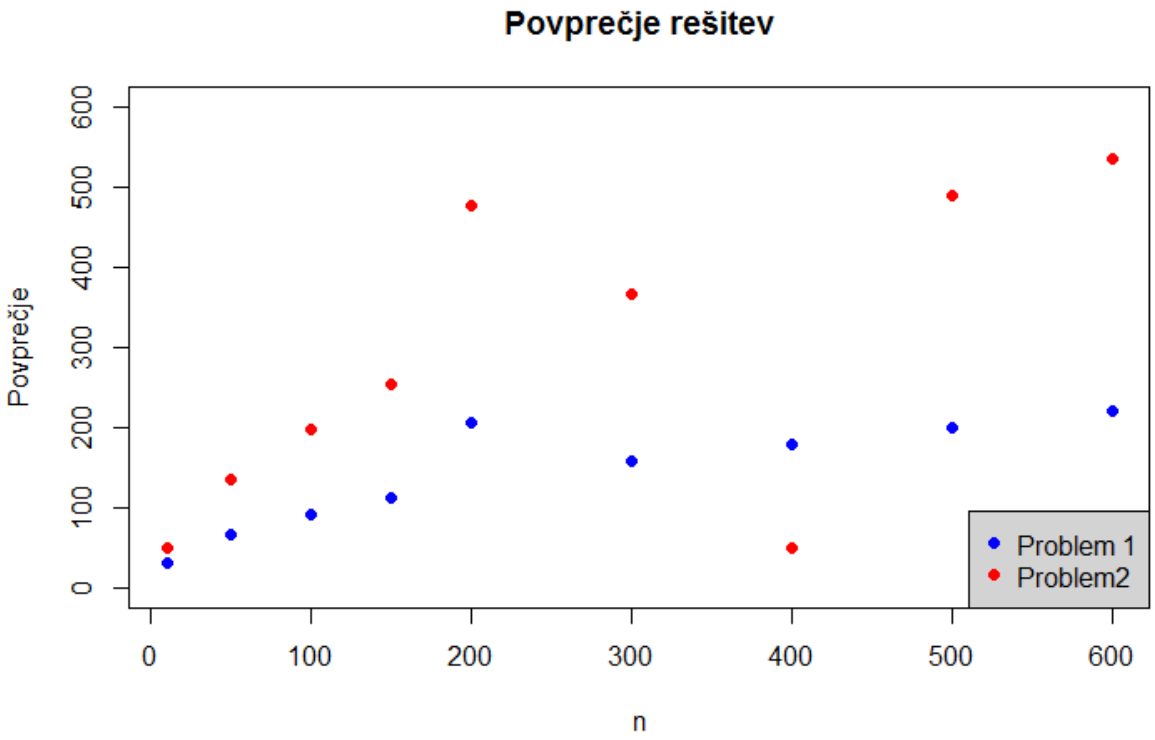
UGOTOVITVE

Odvisnost rešitve od velikosti n

Ugotovili sva, da se vrednost rešitve z večanjem parametra n veča. Ker so točke bolj na gosto posejane v ravnini, se sicer dolžine daljic med dvema najbližjima točkama krajšajo, vendar pa se veča število teh daljic, zaradi česar je tudi skupna dolžina vseh daljic večja.

Preverili sva, kako se spreminjata rešitvi problema 1 in 2 če za neko fiksno število točk spremenimo le njihove lokacije. Program sva izvajali za 10, 50, 100, 150, 200, 300, 400, 500 in 600 točk. Ugotovili sva, da se s številom ponovitev vrednost rešitve problema 1 ne spreminja prav dosti, medtem ko se rešitev problema 2 kar precej spreminja. Z večanjem n pa se veča tudi velikost rešitve pri obeh problemih.

Za natančnejše ugotovitve in bolj pravilne rezultate pa bi morale narediti še več ponovitev. Vidimo, na primer, da je pri n=400 vrednost problema 2 manjša od vrednosti problema 1, kar je precej čudno in bi pričakovali, da bo, tako kot pri ostalih n-jih, vrednost problema 2 večja.

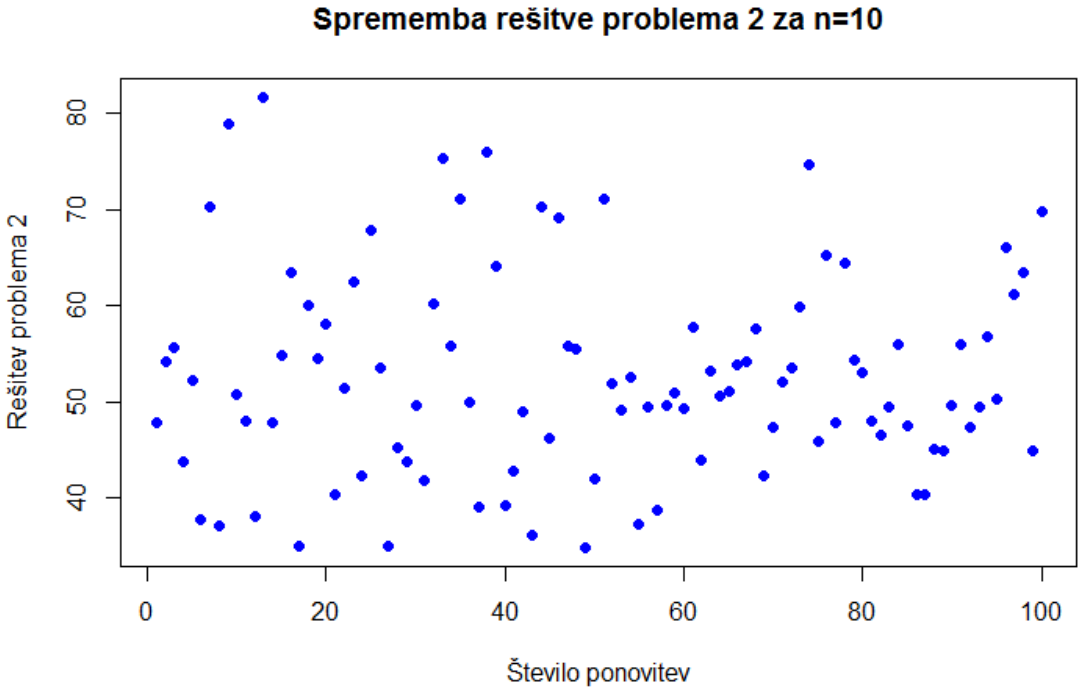


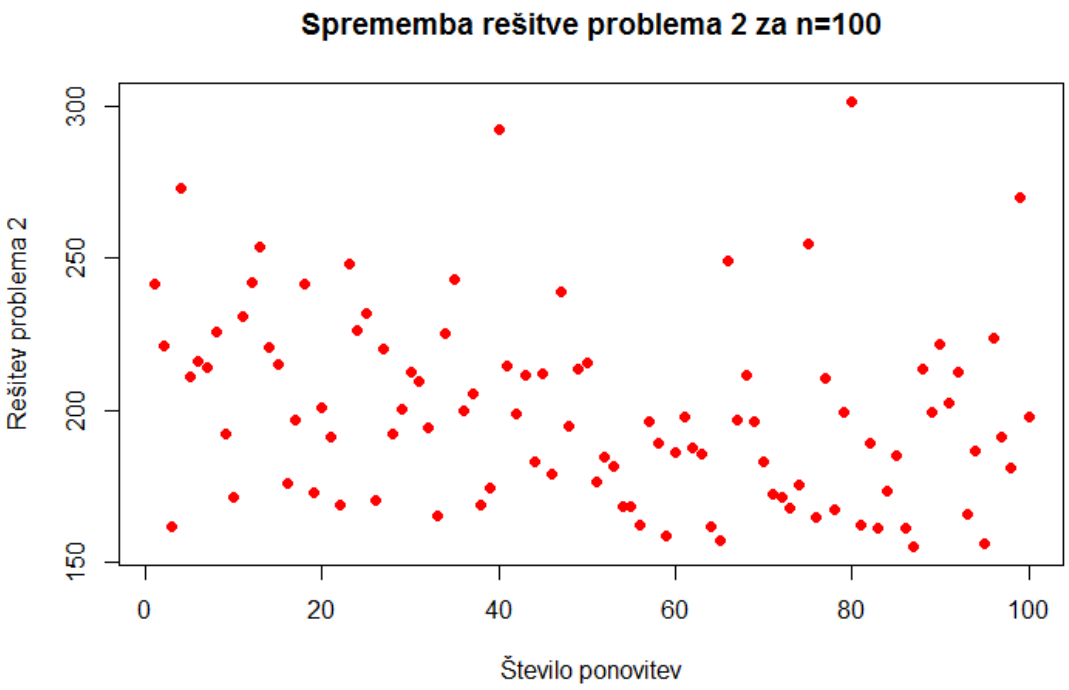
Primerjava rešitev pri problemu 1 in problemu 2

Kot sva predvidevali, je vrednost rešitve pri problemu 1 manjša kot vrednost rešitve pri problemu 2. To je zaradi tega, ker se pri problemu 2 točka ne more povezati direktno s svojo najbližjo točko, ampak se poveže z najbližjo točko iste barve. Zato so vse daljice daljše ali kvečjemu enako dolge kot daljice pri problemu 1. Zaradi tega je skupna dolžina daljic tudi daljša.

Prav tako sva ugotovili, da je skupna dolžina daljic pri problemu 2 bistveno daljša kot pri problemu 1. To najbrž lahko pripišemo temu, da imamo polovico manj potencialnih najbližjih sosednjih točk za posamezno točko. Na prejšnjem grafu lahko vidimo razliko vrednosti teh dveh problemov.

Preverili sva tudi, kako se spreminjajo vrednosti problema 2, če iste točke večkrat različno pobarvava. Ugotovili sva, da so rezultati precej različni. Naredili sva simulacijo za 10 in za 100 točk ter ugotovili, da je pri 10 točkah varianca bistveno manjša kot pri 100 točkah. Prav tako sva ugotovili, da je pri 10 točkah povprečna rešitev problema 2 bistveno bližje rešitvi problema 1 kot pri 100 točkah. Spodnja grafa prikazujeta spremembo rešitve problema 1 za 10 in 100 točk.


Sprememba rešitve problema 2 za n=10
Rešitev problema 2
40
50
60
70
80
0
20
40
60
80
100
Število ponovitev


Sprememba rešitve problema 2 za n=100
Rešitev problema 2
150
200
250
300
0
20
40
60
80
100
Število ponovitev