Univerza v Ljubljani
Fakulteta za matematiko in fiziko

BOTTLENECK ASSIGNMENT IN THE PLANE

Projekt pri predmetu Finančni praktikum

Avtorici:
Lea Holc, Eva Rudolf

Ljubljana, 2023

Kazalo

1 Navodilo

Naj bosta P in Q dani množici točk v ravnini. Elementi množice P predstavljajo proizvajalce (*providers*), elementi množice Q pa porabnike (*clients*). Tako proizvajalci kot porabniki upravljajo z isto dobrino. Vsak proizvajalec $p \in P$ lahko proizvede $s_p > 0$ dobrine, za vsakega porabnika $q \in Q$ pa definiramo njegovo povpraševanje po dobrini kot $d_q > 0$. Predpostavimo, da je ponudba vseh prozvajalcev enaka povpraševanju vseh porabnikov. Cilj projekta je ugotoviti, ali za podano vrednost λ proizvajalci lahko zadostijo potrebam porabnikov in sicer tako, da vsaka dobrina prepotuje pot največ λ.

2 Linearni program

Ukvarjamo se s problemom maksimalnega pretoka, ki ga želimo predstaviti kot linearni program (LP). Cilj je bil napisati linearni program, ki minimizira največjo razdaljo, ki jo mora prepotovati dobrina od proizvajalca do porabnika, pri čemer moramo zadostiti vsem potrebam, ki jih imajo porabniki, ne da bi presegli kapacitete, ki jih lahko proizvede posamezen proizvajalec.

2.1 Oznake

Najprej uvedemo oznake. Naj bo p_i za $i = 1, \ldots, m$ posamezen proizvajalec s koordinatami (x_i, y_i) in s_i za $i = 1, 2, \ldots, m$ njegova proizvodnja dobrine. Naprej naj bo q_j za $j = 1, 2, \ldots, n$ posamezen porabnik s koordinatami (w_j, z_j) in d_j za $j = 1, 2, \ldots, n$ njegovo povpraševanje po dobrini. Spomnimo se, da tako proizvajalci kot porabniki upravljajo z isto dobrino in predpostavimo, da velja

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j.$$

S c_{ij} označimo količino, ki jo proizvajalec i dobavi porabniku j. Ker vsi proizvajlci ne dobavljajo vsem porabnikom, uvedemo še indikatorsko spremenljivko b_{ij}.

$$b_{ij} = \begin{cases} 0; & \text{i-ti proizvajalec ne oskrbuje j-tega ponudnika} \\ 1; & \text{i-ti proizvajalec oskrbuje j-tega ponudnika} \end{cases}$$

Z λ označimo maksimalno razdaljo, ki jo lahko prepotuje posamezna dobrina. To razdaljo želimo minimizirati.

2.2 Pogoji

Zapišimo še pogoje, s katerimi zadostimo potrebam posameznega porabnika in zmožnostim posameznega proizvajalca:

$$\sum_{j=1}^{n} c_{ij} = s_i \text{ za } \forall i = 1, \ldots, m$$

$$\sum_{i=1}^{m} c_{ij} = d_j \text{ za } \forall j = 1, \ldots, n$$

Za indikatorsko spremenljivko b_{ij} lahko zapišemo

$$0 \leq b_{ij} \leq 1 \text{ in } b \in \mathbb{Z}.$$

Veljati mora tudi $b_{ij} = 0 \Rightarrow c_{ij} = 0$, kar lahko zapišemo kot

$$0 \leq c_{ij} \leq c_{ij} \cdot b_{ij}.$$

Omejimo še λ:

$$b_{ij} \cdot \sqrt{(x_i - w_j)^2 + (y_i - z_j)^2} \leq \lambda.$$

Naš cilj bo poiskati

$$\min \max_{i,j} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \cdot c_{ij}.$$

3 Reševanje problema

3.1 Programiranje rešitev

Glede na to, da se ukvarjamo z linearnim programiranjem, sva za zapis algoritmov uporabili programski jezik *SageMath* v okolju *CoCalc*. Napisali sva funkcije za generiranje točk, linearni program in dodatne funkcije za določitev pričakovane vrednosti λ in za analizo časovne zahtevnosti.

Najprej sva definirali tri različne funkcije za generiranje množic točk proizvajalcev P in porabnikov Q. To so funkcije *kvadrat*, *krog* in *strani*, ki se – kot nam povedo njihova imena – razlikujejo glede na to, v katerem območju generirajo točke. Vsaka funkcija sprejme moč množice P, moč množice Q in velikost območja (npr. polmer kroga pri funkciji *krog*).

Vsaka izmed teh funkcij za vsakega proizvajalca oz. točko iz P določi, koliko proizvede, ter za vsakega porabnika oz. točko iz Q, koliko porabi.

Naslednja funkicija, ki sva jo napisali, je funkcija *program*, ki sprejme množici točk, ki jih vrnejo funkcije za generiranje točk. Ta funkcija deluje kot linearen program, ki minimizira največjo razdaljo, ki jo mora prepotovati dobrina od proizvajalca do porabnika, pri čemer moramo zadostiti vsem pogojem, ki smo jih navedli v točki 2.2. Vrne vrednost λ (v programski kodi je to spremenjivka z), množico povezav in uteži povezav; torej kolikšna količina dobrine potuje od proizvajalca i do porabnika j.

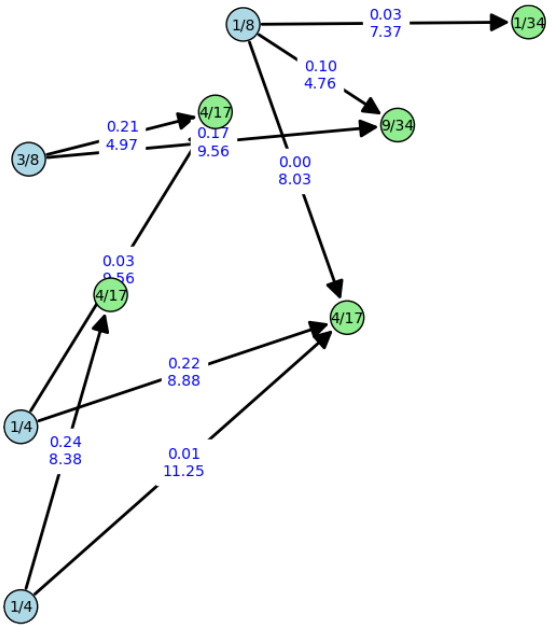
Programska koda za funkcijo *program*:

```
def program(s,d):
m = len(s)
n = len(d)
p = MixedIntegerLinearProgram(maximization=False)
b = p.new_variable(binary=True)
c = p.new_variable(nonnegative=True)
p.set_objective(p['l'])
p.add_constraint((p['l']) >= 0)
for i, ((sx, sy), sk) in enumerate(s):
    for j, ((dx, dy), dk) in enumerate(d):
        p.add_constraint(b[i, j] * sqrt((dx-sx)^2 + (dy-sy)^2) <= p['l'])
        p.add_constraint(c[i, j] >= 0)
        p.add_constraint(c[i, j] <= sk * b[i, j])
for i, (sxy, sk) in enumerate(s):
    p.add_constraint(p.sum(c[i, j] for j in range(n)) == sk)
for j, (dxy, dk) in enumerate(d):
    p.add_constraint(p.sum(c[i, j] for i in range(m)) == dk)
z = p.solve()
povezave = p.get_values(b)
kolicine = p.get_values(c)
return z, povezave, kolicine
```

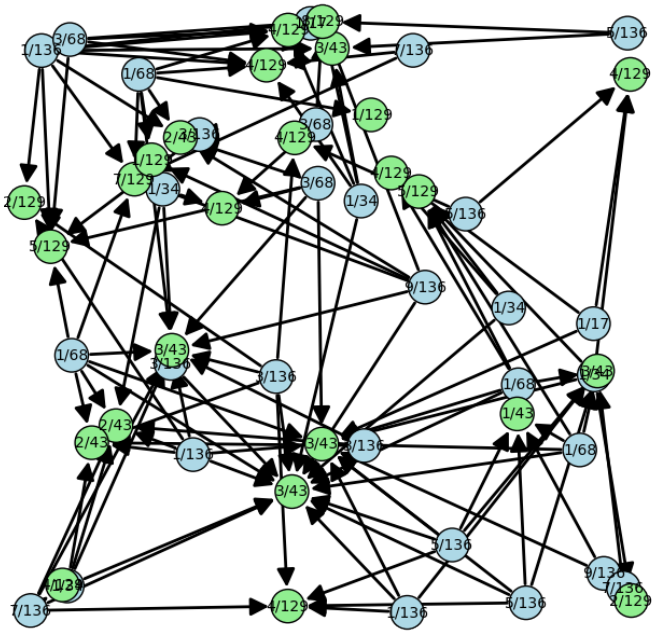
3.2 Grafični prikaz rezultatov

Na spodnjih grafih so prikazane rešitve za različne parametre in različne načine generiranja točk. Na grafih z manj točkami so prikazane tudi uteži povezav. Le-ti sta na vsaki povezavi dve – zgornja pomeni količino, ki jo proizvajalec p_i dostavi porabniku q_i, spodnja pa je razdalja med vozliščema. Na grafih z več točkami so zaradi preglednosti uteži povezav izpuščene.

Če za generiranje točk uporabimo funkcijo *kvadrat*, dobimo naslednja dva grafa:

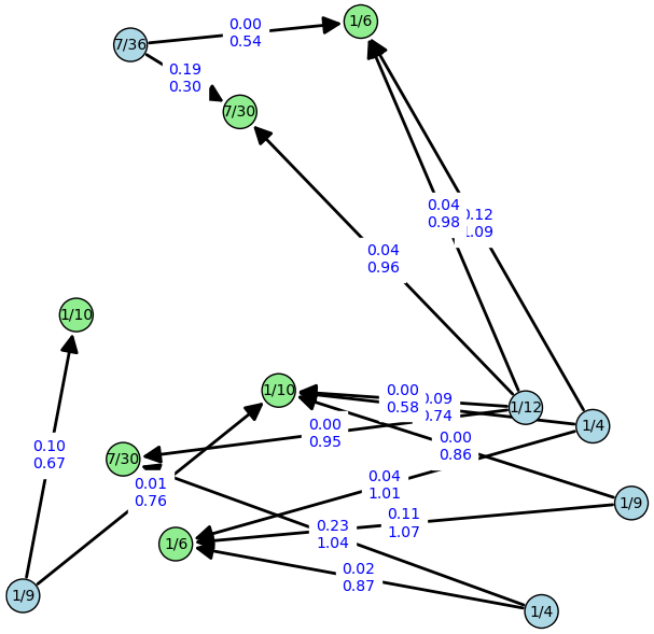


Slika 1: Generiranje točk v kvadratu s stranico 20 za $|P| = 4$, $|Q| = 5$

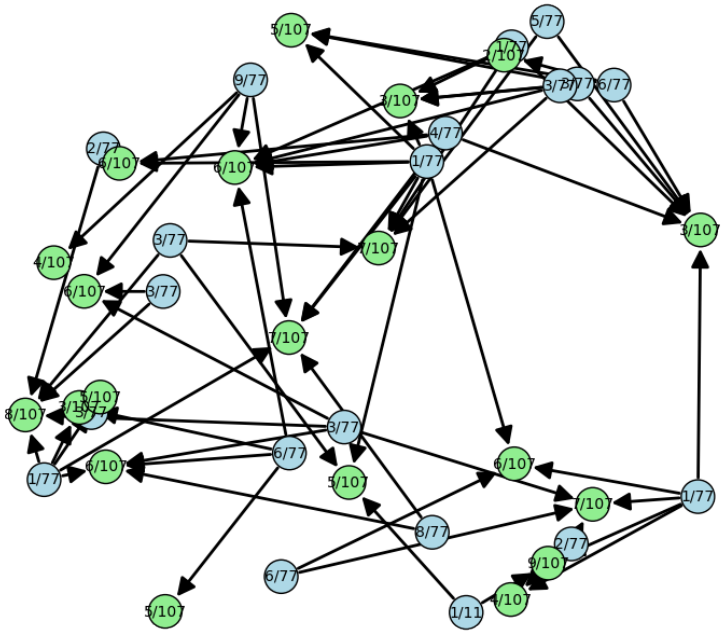


Slika 2: Generiranje točk v kvadratu s stranico 40 za $|P| = 30$, $|Q| = 25$

Če uporabimo funkcijo *krog*:

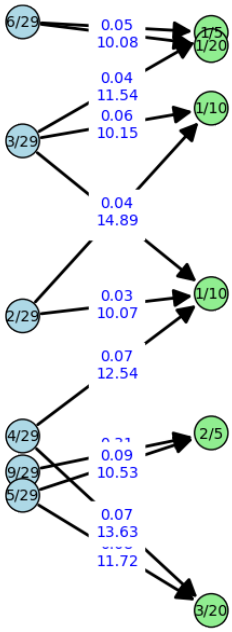


Slika 3: Generiranje točk v krogu s polmerom 1 za $|P| = 6$, $|Q| = 6$

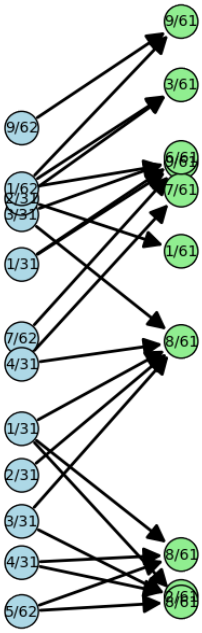


Slika 4: Generiranje točk v krogu s polmerom 3 za $|P| = 20$, $|Q| = 20$

Za konec generiramo še točke na dveh vzporednih daljicah s pomočjo funkcije *stani*:



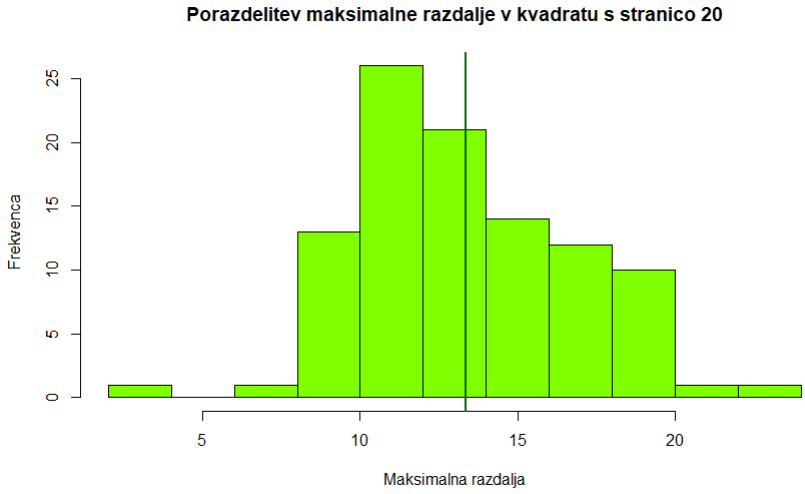
Slika 5: Generiranje točk na vzporednih daljicah za $|P| = 6$, $|Q| = 6$

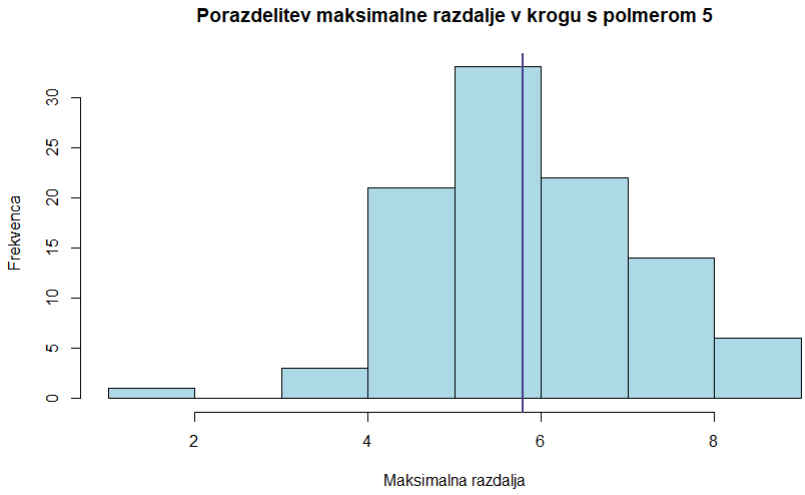


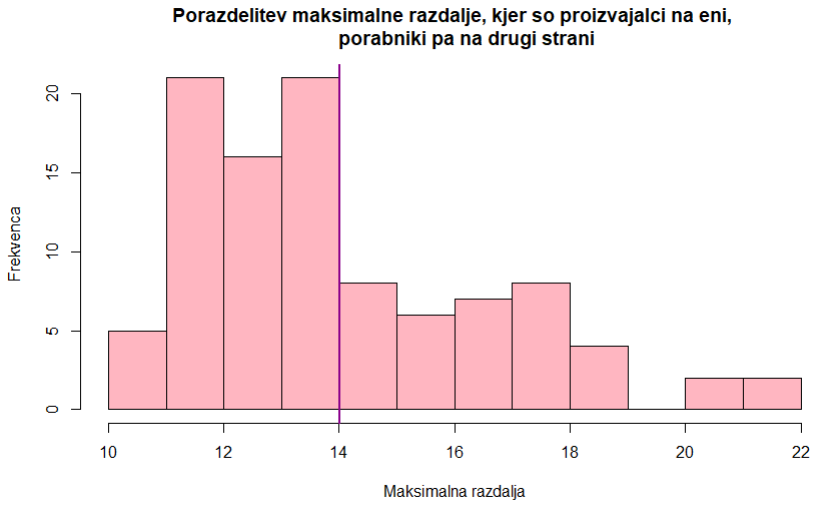
Slika 6: Generiranje točk na vzporednih daljicah za $|P| = 12$, $|Q| = 10$

4 Analiza

Da bi analizirali pričakovano vrednost razdalje λ, sva večkrat zagnali funkcijo *program* s pomočjo funkcije *simulacija*. Rezultate sva shranili v `.csv` datoteko in s pomočjo programa `R` izrisali še histograme, ki kažejo pričakovane vrednosti λ za izbrane parametre v odvisnosti od tega, kje generiramo točke, ki predstavljajo proizvajalce in porabnike. Rezultati so prikazani na spodnjih grafih.






Porazdelitev maksimalne razdalje, kjer so proizvajalci na eni, porabniki pa na drugi strani
Frekvenca
0
5
10
15
20
10
12
14
16
18
20
22
Maksimalna razdalja