Univerza v Ljubljani
Finančna matematika 1. stopnja

Najcenejše prirejanje v ravnini

Finančni praktikum

Iza Čebulj, Barbara Pal

Ljubljana, 2023

Kazalo

1 Reševanje osnovnega problema najcenejšega prirejanja

1.1 Opis problema

Množici P z $2n$ točkami lahko priredimo neusmerjen graf $G(P, E)$. Množica vozlišč grafa G je kar množica P, množica povezav v grafu E pa so neurejeni pari vozlišč (u, v), za katere velja $u,\ v \in P$ in $u \neq v$. Cena povezave je razdalja $d(u, v)$ med vozliščema u in v.
Popolno prirejanje na grafu G oziroma v množici P je taka množica povezav M, za katero velja, da vsako vozlišče v P sovpada z natanko eno povezavo v M. Ceno prirejanja definiramo kot $\sum_{(u,v)\in M} d(u, v)$, kar je vsota cen vseh povezav v M.
Radi bi poiskali *najcenejše popolno prirejanje* in njegovo ceno za različne n.

1.2 Celoštevilski linearni program in programiranje rešitev

Pri svojem delu sva uporabljali spletno platformo *CoCalc,* ki omogoča urejanje *Jupyter* dokumentov. Algoritem za iskanje najcenejšega prirejanja sva napisali v sistemu *SageMath,* ki uporablja podobno sintakso kot *Python.* Najprej sva definirali funkcijo, ki nama je generirala $2n$ točk v enotskem kvadratu.

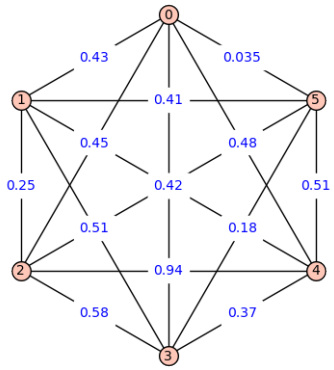
```
def generiranje_tock_kvadrat(n):
V = RDF^2 # vektorski prostor R^2 (ravnina)
tocke = [V.random_element(min=0, max=1) for _ in range(2*n)]
return tocke
```

Podobno sva definirali še funkciji, ki generirata točke v enotskem krogu in v enakostraničnem trikotniku.

Za tem pa so generirane točke predstavljale vozlišča grafa, sam graf pa je bil poln, in tako lahko algoritem pri iskanju najkrajše razdalje oziroma najnižje cene upošteva vse povezave. V definicijo za generiranje grafa sva torej vključili ukaze za n, *generiranje točk* in *normo,* s katerimi lahko izberemo število $2n$ vozlišč, lik, iz katerega jih poberemo, ter normo, po kateri izračunamo razdaljo. Privzeta vrednost norme je 2, torej evklidska norma oziroma običajna razdalja med dvema točkama, seveda pa funkcija `graf` upošteva tudi normi 1 in *Infinity.*

```
def graf(n, generiranje_tock, norma=2):
    tocke = generiranje_tock(n)
    G = graphs.CompleteGraph(len(tocke))
    for u, v in G.edges(labels=False):
        G.set_edge_label(u, v, (tocke[u] - tocke[v]).norm(norma))
    return G
```

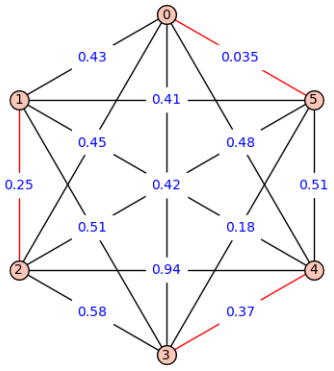
Definirali sva tudi funkcijo, ki graf izriše s približki cen povezav. Tako izgleda generiran in izrisan graf na 6 točkah, naključno izbranih v enotskem kvadratu, cene na povezavah pa so evklidske razdalje med vozlišči.



Končno pa sva s pomočjo *SageMath-a* napisali še **celoštevilski linearni program**, ki vrne pare vozlišč, med katerimi so povezave v najcenejšem prirejanju M, vsoto cen povezav v M in nariše graf z označenimi povezavami, ki so v najcenejšem prirejanju.

```
def clp(G):
    p = MixedIntegerLinearProgram(maximization=False)
    b = p.new_variable(binary=True)
    p.set_objective(sum([w * b[Set(e)] for *e, w in G.edges(labels=True)]))

    for v in G:
        p.add_constraint(sum([b[Set(e)]
                        for e in G.edges_incident(v, labels=False)]) == 1)

    cena = p.solve()
    b = p.get_values(b)

    M = [tuple(e) for e, i in b.items() if i]
    print(M) # pari vozlišč, med katerimi so povezave v najcenejšem prirejanju

    x = [w for *e, w in G.edges() if tuple(e) in M] # seznam cen povezav v M
    print(sum(x)) # vsota cen povezav v M

    H = Graph([(*e, N(w, digits=2)) for *e, w in G.edges(labels=True)])
    H.set_pos(G.get_pos())

    return H.plot(edge_colors={"red": M}, edge_labels=True)
    # graf H z rdeče pobarvanimi povezavami iz prirejanja
```

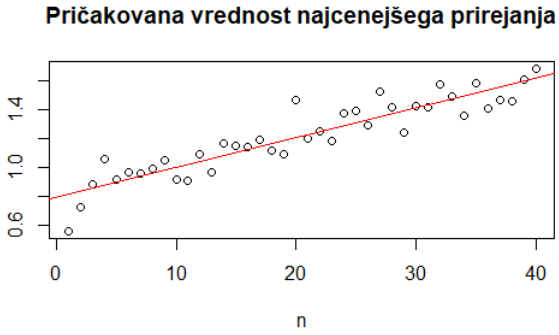
Ko na generiranem grafu želimo najti povezave v najcenejšem prirejanju z uporabo funkcije `clp` dobimo seznam parov vozlišč, med katerimi so povezave v najcenejšem prirejanju: $[(0,5),(1,2),(3,4)]$ in skupno vsoto cen teh povezav 0.6531541582221374 (slika 1).



Slika 1: Graf z označenimi povezavami, ki so v najcenejšem prirejanju

1.3 Analiza rezultatov

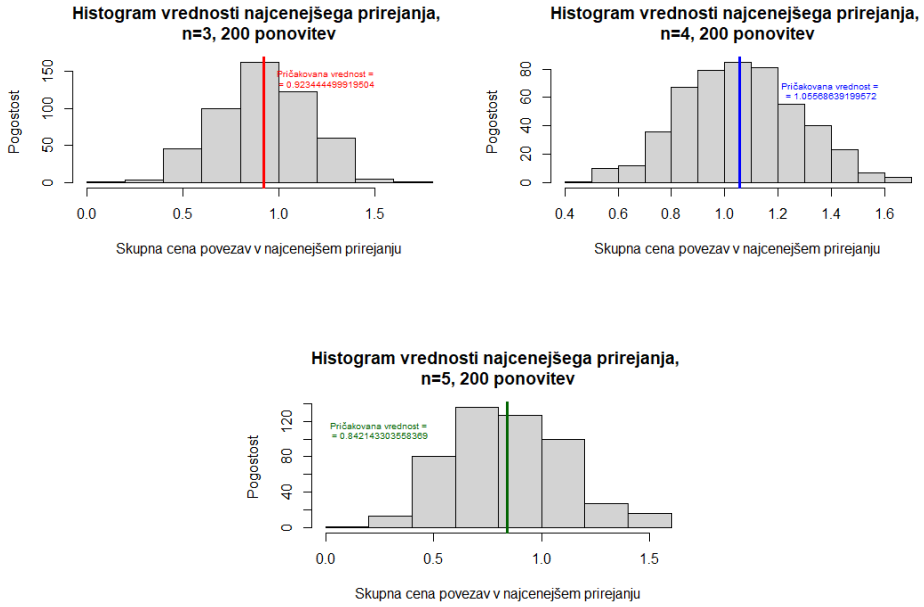
Celoštevilski linearnih program sva večkrat pognali na generiranih grafih, dobljene skupne vsote cen povezav v najnižjem prirejanju pa sva izpisovali v `.csv` datoteko za lažji uvoz in obdelavo v programskem jeziku *R*. Nato sva iz vektorja večih ponovitev izračunali njegovo povprečje za posamezen n. Najprej naju je zanimalo, kaj se dogaja s pričakovano vrednostjo najcenejšega prirejanja v polnem grafu, ko točke izberemo enakomerno v enotskem kvadratu.



Slika 2: Graf pričakovane vrednosti najcenejšega prirejanja v odvisnosti od n, za $n = 1, 2, \ldots, 40$.

Na grafu 2 lahko vidimo, da se z n povečujejo tudi pričakovane vrednosti najcenejših prirejanj. Zanimivo sa nama je zdelo predvsem, da po začetnem linearnem naraščanju pri $n = 1, 2, 3$ in 4, pričakovana vrednost pade, preden začne spet naraščati. Pade spet pri $n = 10$ in še nekajkrat za večje n. Odstopanja od linearnosti sva pripisali malemu številu ponovitev (zaradi počasnosti algoritma sva CLP pognali 10-krat), sva pa se odločili za nizke

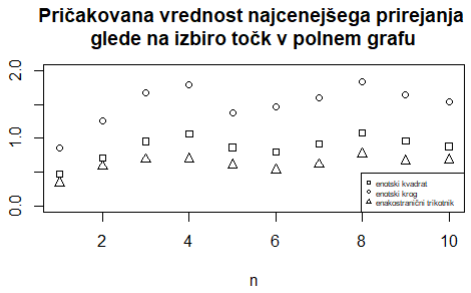
$n = 3, 4, 5$ algoritem ponoviti dvestokrat.



Kot je razvidno iz histogramov, se z $n = 3$ na $n = 4$ pričakovana vrednost skladno z linearnostjo poveča, pri $n = 5$ pa tudi pri 200 ponovitvah zmanjša.

Izbira točk in norme

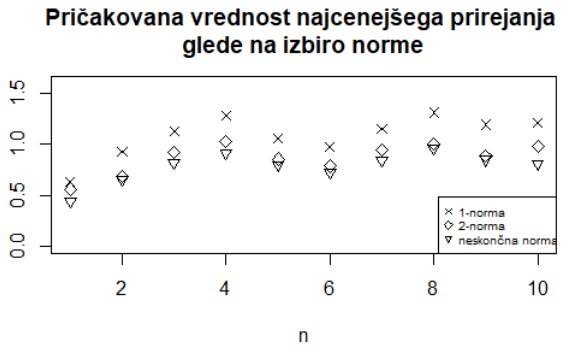
Pri generiranju grafov sva lahko izbirali med grafi, ki izberejo točke enakomerno v enotskem kvadratu, enotskem krogu in enakostraničnem trikotniku.



Slika 3: Primerjava pričakovanih vrednosti cene najcenejšega prirejanja glede na izbiro točk

S slike 3 je lepo razvidno, da je cena povezav (za n od 1 do 10) v prirejanju najnižja, ko jih izberemo v enakostraničnem trikotniku in najvišja, ko jih izberemo v enotskem krogu.

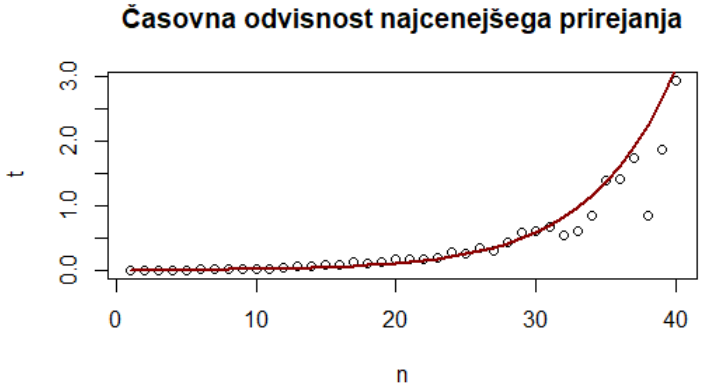
V funkcijo za generiranje grafov oziroma natančneje za izračun dolžin med točkami, tj. cen povezav, sva uvedli tudi možnost izračuna z različnimi normami. Na sliki 4 vidimo, da bo popolno prirejanje imelo najnižjo ceno, če razdaljo med točkami računamo z neskončno normo in najvišjo, če jo računamo z 1-normo.



Slika 4: Primerjava pričakovanih vrednosti cene najcenejšega prirejanja glede na izbiro norme

Časovna zahtevnost algoritma

Časovna zahtevnost algoritma, ki vrne najcenejše prirejanje, je v najinem primeru *eksponentna*. Za večje n je bilo vedno zahtevneje najti prirejanje z najnižjo ceno, zato sva pričakovane vrednosti poiskali za graf z največ 80 vozlišči ($n = 40$).



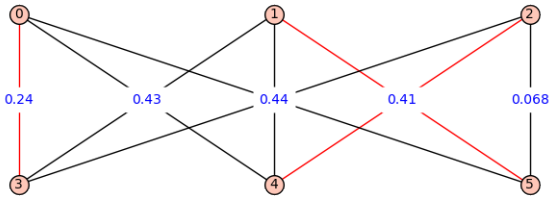
2 Dvobarvno najcenejše prirejanje

2.1 Opis problema

Množico P sestavljata množica n rdečih točk, R, in množica n modrih točk B, $P = R \cup B$. V tem primeru je $G(P, E)$ dvodelen graf z lastnostjo, da med dvema točkama obstaja povezava, če in samo če sta različnih barv. Cene povezav (u, v) so tako kot v osnovnem primeru razdalje med vozlišči, $d(u, v)$. Spet iščemo najcenejše popolno prirejanje in njegovo ceno.

2.2 Programiranje rešitev in eksperimentiranje

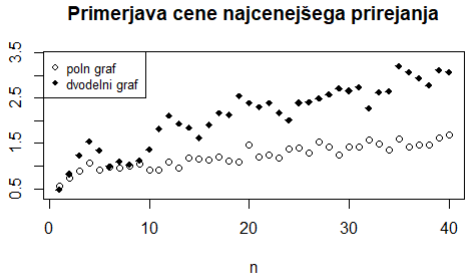
Za programiranje rešitev sva ponovno uporabili *SageMath*. Napisali sva funkcijo `dvobarven_graf`, ki je precej podobna prvotni funkciji, razlikuje se predvsem v tem, da graf izdela s funkcijo `CompleteBipartiteGraph(n, n)`. Za iskanje najcenejšega prirejanja sva uporabili enak celoštevilski linearni program.



Slika 5: Rešen problem najcenejšega prirejanja na dvobarvnem grafu, ko točke izberemo naključno v enakostraničnem trikotniku in za računanje razdalje med njimi uporabimo neskončno normo.

2.3 Analiza rezultatov

Kot je razvidno s spodnje slike, tudi v dvodelnem grafu pričakovana vrednost najcenejšega prirejanja narašča, vendar pa ne narašča enako hitro kot v polnem grafu. Že pri $n = 12$ je razlika v skupni ceni, ko točke izberemo na enotskem kvadratu, večja od ene enote.



Pri **časovni odvisnosti** algoritma lahko vidimo, da v dvodelnem grafu najcenejše prirejanje najdemo veliko hitreje kot v polnem. Razlika je vse bolj očitna z večanjem števila vozlišč. Temu je verjetno tako, ker ima dvobarvni graf manj povezav, ki bi jih moral CLP upoštevati pri iskanju prirejanja.