Univerza v Ljubljani
Finančna matematika 1. stopnja

Najcenejše prirejanje v ravnini

Finančni praktikum

Iza Čebulj, Barbara Pal

Ljubljana, 2022

Kazalo

1 Reševanje osnovnega problema najcenejšega prirejanja 3

1.1 Opis problema 3

1.2 Zapis problema kot celoštevilski linearni program 3

1.3 Programiranje rešitev in eksperimentiranje 3

1.4 Analiza rezultatov 5

2 Dvobarvno najcenejše prirejanje 5

2.1 Opis problema 5

2.2 Programiranje rešitev in eksperimentiranje 6

2.3 Analiza rezultatov 6

3 Zaključek 6

Slike

1 Graf z označenimi povezavami, ki so v najcenejšem prirejanju 5

2 Rešen problem najcenejšega prirejanja na dvobarvnem grafu, ko točke izberemo naključno v enakostraničnem trikotniku in za računanje razdalje med njimi uporabimo neskončno normo. 6

1 Reševanje osnovnega problema najcenejšega prirejanja

1.1 Opis problema

Množici P z $2n$ točkami lahko priredimo neusmerjen graf $G(P, E)$, oziroma samo G. 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. Velikost popolnega prirejanja v množici velikosti $2n$ je n. 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 Zapis problema kot celoštevilski linearni program

1.3 Programiranje rešitev in eksperimentiranje

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
```

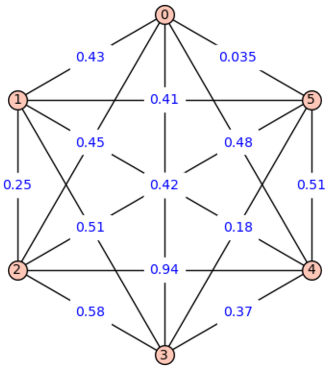
Podobno sva definirali še funkciji, ki generirata točke v enotskem krogu in v enakostraničnem trikotniku.

Za tem pa sva želeli, da generirane točke predstavljajo vozlišča grafa, sam graf pa bo poln, torej so vse točke povezane med sabo in tako lahko algoritem pri iskanju najkrajše razdalje oziroma najnižje cene upošteva vse možne razdalje oziroma cene. V definicijo za generiranje grafa, kjer cene na povezavah predstavljajo razdaljo med točkami, sva torej vključili ukaze za n, *generiranje tock* in *normo,* s katerimi lahko izberemo število $2n$ točk, lik, iz katerega jih poberemo, ter normo, po kateri izračunamo razdaljo. Privzeta vrednost norme je 2, torej je to 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.

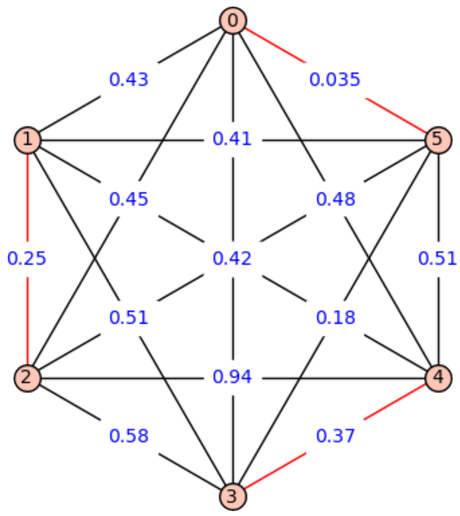


Končno pa sva s pomočjo *SageMath-a* napisali še algoritem, ki s celoštevilskim linearnim programom reši problem najcenejšega prirejanja, izpiše pare vozlišč, med katerimi so povezave v najcenejšem prirejanju M, vsoto cen povezav v M in nariše graf z označenimi povezavami, ki so v najcenejšem prirejanju.

```
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 dobljenem grafu želimo najti povezave v najcenejšem prirejanju pa 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

1.4 Analiza rezultatov

Celoštevilski linearnih program sva večkrat pognali na različnih grafih, dobljene skupne vsote cen v najnižjem prirejanju pa sva izpisovali v `.csv` datoteko za lažji uvoz in obdelavo v programskem jeziku R.

Časovna odvisnost algoritma

2 Dvobarvno najcenejše prirejanje

2.1 Opis problema

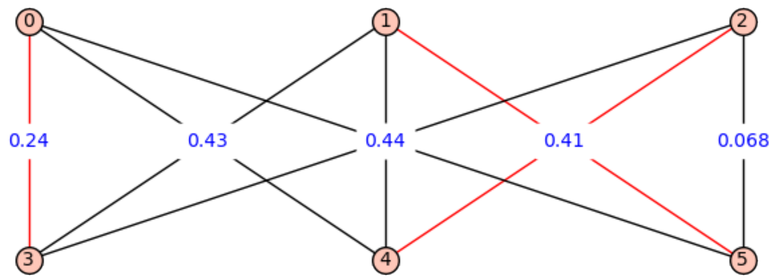
Množico P sestavljata množica n rdečih točk, R, in množica n modrih točk B, $P = R \cup B$. V tem primeru je $G(P,E)$ dvodelen graf z lastnostjo, da med dvema točkama obstaja povezava, če in samo če sta različnih barv. Cene povezav (u, v) so tako kot v osnovnem primeru razdalje med vozlišči, $d(u, v)$. 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, prav tako pa sprejme ukaze za število točk, način generiranja točk oziroma lik in vrsto norme.

```
def dvobarven_graf(n, generiranje_tock, norma):
    G = graphs.CompleteBipartiteGraph(n, n)
    tocke = generiranje_tock(n)

    for u, v in G.edges(labels=False):
        G.set_edge_label(u, v, (tocke[u] - tocke[v]).norm(norma))
    return G
```

Za iskanje najcenejšega prirejanja pa sva uporabili enak celoštevilski linearni program.



Slika 2: 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

3 Zaključek