Univerza v Ljubljani
Fakulteta za matematiko in fiziko

MANY SHORT PATHS

Finančni praktikum

Tim Dolenc, Jošt Gojkovič

Ljubljana, 2022

1 Predstavitev problema

1.1 Navodilo

Za vsako pozitivno celo število k naj bo P_k pot s k povezavami in $k+1$ vozlišči. Za graf G in pozitivno celo število k, naj bo $\phi(G,k)$ število kopij poti P_k v G, to je, število podgrafov grafa G ki so izomorfni P_k. Podobno, naj bo $\phi_i(G,k)$ število induciranih kopij poti P_k v G. Izračunajte $\phi(G,k)$ in $\phi_i(G,k)$ za graf G nekaterih družin grafov za majhne vrednosti k.

1.2 Definicije

Definicija 1. *Izomorfizem grafov G in H je takšna bijektivna preslikava med množicama vozlišči grafa G in H:*

$$f : V(G) \to V(H)$$

da sta poljubni dve vozlišči u in v garfa G sosednji v G, če in samo če sta $f(u)$ in $f(v)$ sosednji v H. Če med dvema grafoma G in H obstaja izomorfizem, sta grafa med seboj izomorfna, kar se označi kot $G \simeq H$.

Definicija 2. *Naj bo $G = (V, E)$ poljuben graf in $S \subset V$ poljubna podmnožica vozlišč grafa G. Potem je $G[S]$ induciran podgraf katerega množica vozlišč je S in množica povezav je sestavljena iz vseh povezav v E, ki imajo končna vozlišča v S.*

2 Opis postopka

Za reševanje problema sva uporabila program, napisan v Pythonu, ki za poljuben k iz $\mathbb{N}$ in poljuben G (podan s seznamom sosedov) poišče vse kopije P_k in vse inducirane kopije P_k v grafu G in posledično vrednosti funkcij $\phi(G,k)$ in $\phi_i(G,k)$.

Opis programa:

1. za vsako vozlišče v iz G poiščemo množico poti dolžine k, ki se začnejo v v in so podgrafi G
2. poiščemo M, ki je unija množic iz (1), pri tem upoštevamo, da sta v neusmerjenem grafu poti z istimi vozlišči v obratnem vrstnem redu "ekvivalentni"
3. izračunamo moč M. To nam da vrednosti $\phi(G,k)$
4. za vsak nesoseden par vozlišč u, v iz vsake poti iz M preverimo, ali sta u, v sosednji v G, v primeru da sta, po definiciji (3) sledi, da pot ni inducirana v G, zato jo odstranimo iz M. Moč, na zgornji način prečiščene, M je iskana vrednost $\phi_i(G,k)$

2.1 Algoritem

```
from copy import deepcopy

def vse_pk_iz_v(G, v, k, length=False):

"""vrne seznam vseh 'navadnih' podgrafov G (predstavljenim s sez. sosedov),
ki so kopija Pk in se začnejo v v.
Argumenti:
 - G ... seznam sosedov,
 - v ... začetno vozlišče
 - lenght ... True, če naj funkcija vrne samo število najdenih poti, sicer
 vrne seznam vseh poti. """

poti = [[v]]            #iskan seznam vseh poti iz v

for _ in range(k):      #želimo prehoditi pot dolžine k, zato postopek
                        #ponovimo k-krat

    vse_nove_poti = []  #ustvarimo nov seznam da ne pokvarimo starega
    for pot in poti:
        v = pot[-1]        #zadnji element poti postane nov koren (v)

        nove_poti = []     #nove_poti so poti, ki so nadaljevanje
                           #spremenljivke pot, torej imajo pri npr. k=2
                           #že dva fiksna elementa

        for sosed in G[v]:    #pogledamo vse sosede v-ja
            if sosed not in pot:
                nova_pot = pot + [sosed]  #če sosed še ni element trenutne
                                          #poti, ga dodamo na konec ene izmed
                                          #novo ustvarjenih poti (za vsakega
                                          #ustreznega soseda dobimo novo)

                nove_poti.append(nova_pot)   #novo pot dodamo k novim potem

        vse_nove_poti = vse_nove_poti + nove_poti  #tako najdene poti dodamo
                                                   #vsem novim potem

    poti = vse_nove_poti     #šele na koncu updatamo občutljiv seznam "poti"
if length:
    return len(poti)
else:
    return poti
```

```
def vse_pk(G, k, length=True):

    """fi(G,k), funkcija za poljuben graf G (sez sosedov) in vsako
    njegovo vozlišče v poišče kopije Pk ki se začnejo (ali končajo) v v
    in naredi njihovo unijo"""

    vse = []
    for v in range(len(G)): #iterira čez vozlišča G
        for pot in vse_pk_iz_v(G,v,k,length=False):
            if pot[::-1] not in vse and pot not in vse:     #preverimo,
                                # če je pot v katerem koli vrstnem redu že
                                # vsebovana v seznamu

                vse.append(pot) #če ni, jo dodamo med iskan seznam
    if length:
        return len(vse)
    return vse

def vse_inducirane_pk(G, k, length=True):

    """ fi_i(G,k)
    vrne število induciranih podgrafov izomorfnih Pk, na način
    da za vsak Pk-ju izomorfen podgraf G-ja preveri ali je induciran,
    sicer ga odstrani.
    """

    ustrezne_poti = deepcopy(vse_pk(G,k,length=False))
    for pot in vse_pk(G,k,length=False):
        for u in pot:
            for v in pot:
                if u in G[v] and abs(pot.index(u)-pot.index(v)) != 1:
                    ustrezne_poti.remove(pot) #odstranimo poti, ki vsebujejo
                                        #vozlišča u,v ki v poti nista sosednja,
                                        #v G pa sta
                    break
            if u in G[v] and abs(pot.index(u)-pot.index(v)) != 1: #dvakrat isti
                                #pogoj, da pademo iz dveh notranjih for zank,
                                #ko pot ostranimo
                break

    if length:
        return len(ustrezne_poti)
    return ustrezne_poti
```

2.2 Generacija grafov

Drugi del naloge je bilo generiranje družin grafov, na katerih sva izvajala zgornji postopek. Izbrala sva naslednje družine grafov:

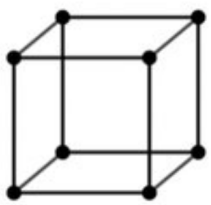
- n-dimenzionalne hiperkocke
- večstopenjske Sierpinski grafe
- platonske grafe (skelet konveksnega poliedra)
- "Circular Ladder Graphs (CLG)"

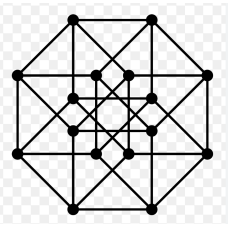
Za generacijo seznama sosedov n-dimenzionalnih hiperkock sva uporabila Python, za ostle tri pa sva uporabila Sage.

3 Analiza rezultatov

3.1 Hiperkocke

Hiperkocke so se nama zdele zanimiva izbira, zaradi dvodelnosti. Posledica dvodelnosti je, da sta funkciji $\phi(G,k)$ in $\phi_i(G,k)$ za k = 2 enaki. Vseeno pa se je za najin ciljni izračun ta družina grafov izkazala za neučinkovito. Zaradi eksponentnega naraščanja števila vozlišč sta tako generacija kot izračun $\phi(G,k)$ in $\phi_i(G,k)$ vzela ogromno časa že za majhne k-je.

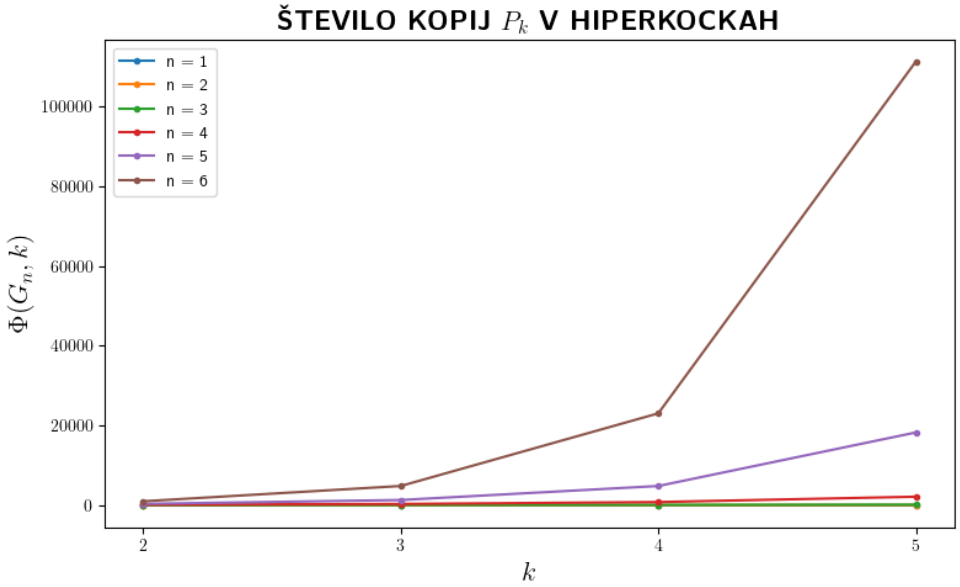


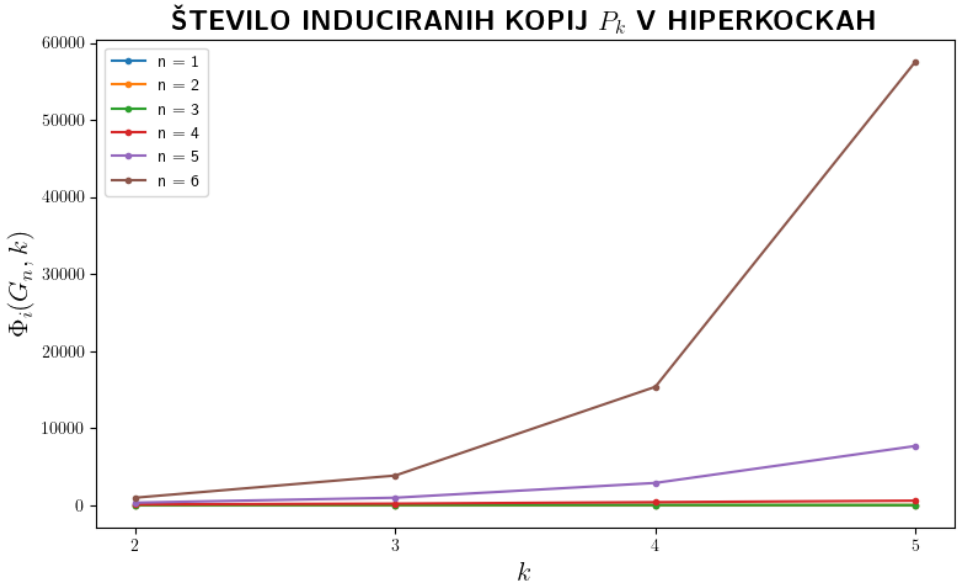


Spodnji tabeli prikazujeta izračune funkcij $\phi(G,k)$ (prva tabela) in $\phi_i(G,k)$ (druga tabela) za hiperkocke dimenzije $n = 1,2,3,4,5,6$ za $k = 2,3,4,5$, kjer je k dolžina poti.

Število kopij P_k						
k	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6
2	0	4	24	96	320	960
3	0	4	48	288	1280	4800
4	0	0	72	768	4800	23040
5	0	0	120	2112	18240	111360

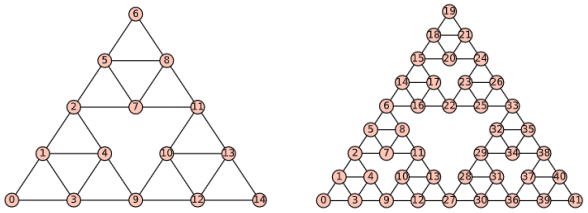
Število induciranih kopij P_k						
k	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6
2	0	4	24	96	320	960
3	0	0	24	192	960	3840
4	0	0	24	384	2880	15360
5	0	0	0	576	7680	57600





3.2 Sierpinski grafi

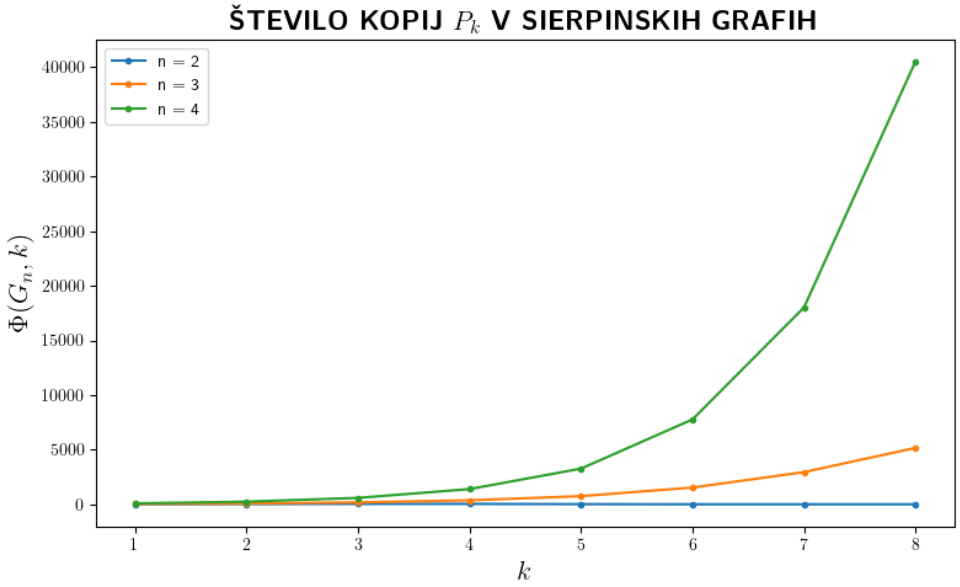
Za generacijo seznama sosedov Sierpinski grafov sva uporabila Sage. Ukaz za generacijo teh grafov je SierpinskiGasketGraph(n). To družino grafov sva izbrala predvsem zaradi estetskih razlogov. Spodaj sta SierpinskiGasketGraph(3) (levo) in SierpinskiGasketGraph(4) (desno).

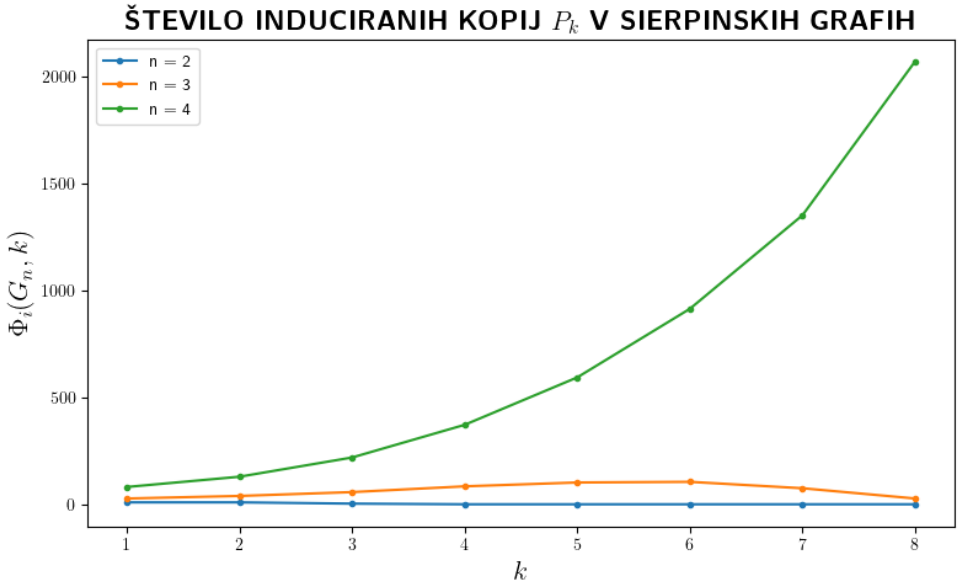


Spodnji tabeli prikazujeta izračune funkcij $\phi(G,k)$ (prva tabela) in $\phi_i(G,k)$ (druga tabela) za Sierpinksi grafe dimenzije $n = 2,3,4$ za $k = 1,\ldots,8$, kjer je k dolžina poti.

Število kopij P_k			
k	n = 2	n = 3	n = 4
1	9	27	81
2	21	75	237
3	33	171	585
4	33	363	1389
5	12	744	3264
6	0	1530	7770
7	0	2943	18021
8	0	5163	40545

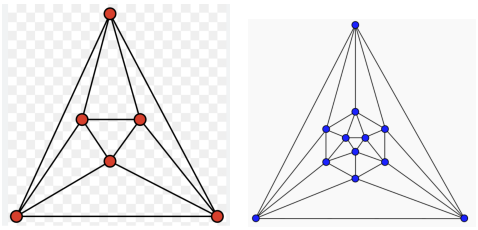
Število induciranih kopij P_k			
k	n = 2	n = 3	n = 4
1	9	27	81
2	9	39	129
3	3	57	219
4	0	84	372
5	0	102	594
6	0	105	915
7	0	75	1353
8	0	27	2073





3.3 Platonski grafi

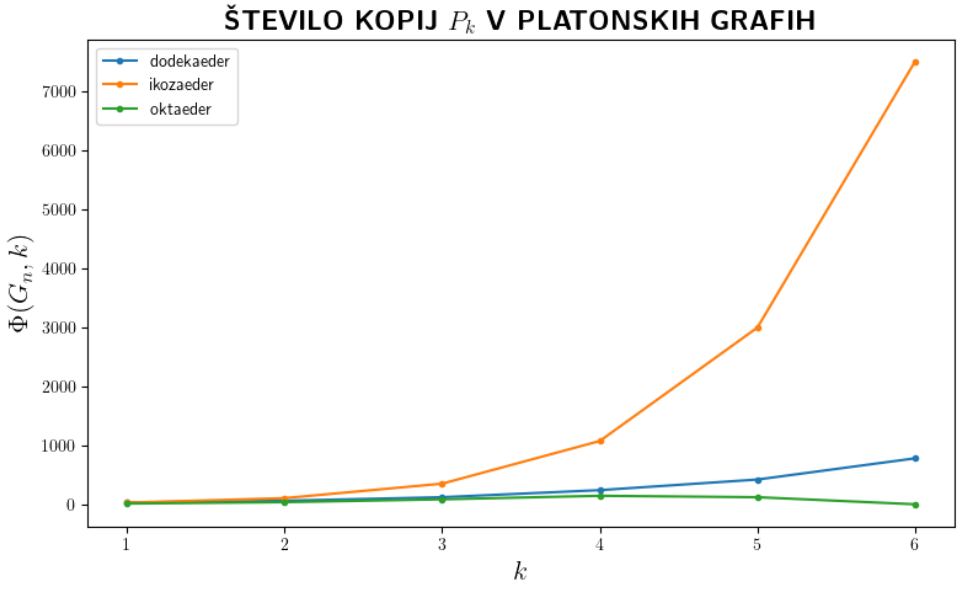
Za generacijo platonskih grafov sva uporabila Sage. Na spodnjih slikah so skeleti oktaedra, dodekaedra in ikozaedra.

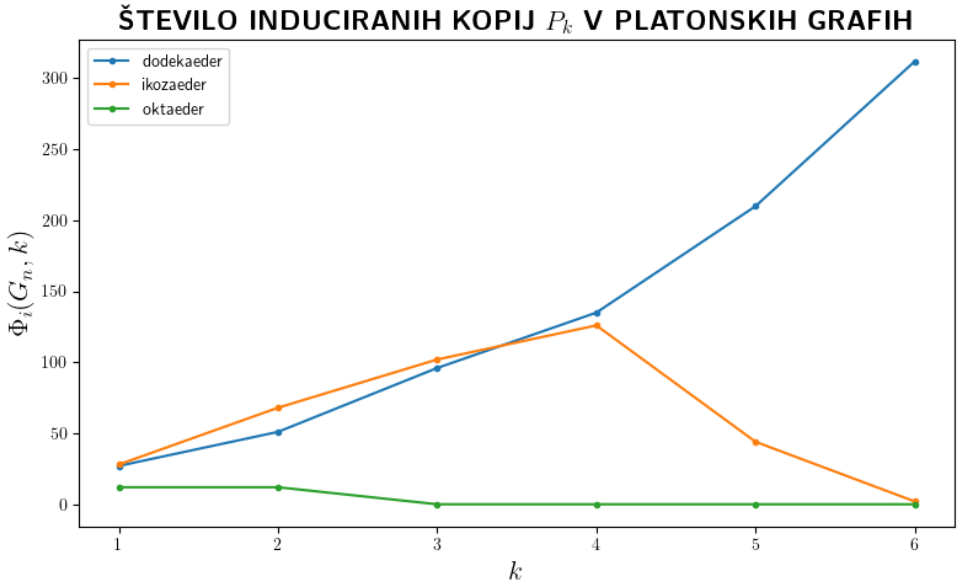


Spodnji tabeli prikazujeta izračune funkcij $\phi(G,k)$ (prva tabela) in $\phi_i(G,k)$ (druga tabela) za izbrane platonske grafe, torej oktaeder, dodekaeder in ikozaeder za $k = 1,\ldots,6$, kjer je k dolžina poti. Čeprav manj kot pri hiperkockah tudi pri tej družini grafov $\phi(G,k)$ in $\phi_i(G,k)$ hitro naraščata in spet podobno opazimo, da je razlika med $\phi(G,k)$ in $\phi_i(G,k)$ vedno večja.

Število kopij P_k			
k	oktaeder	dodekaeder	ikozaedrski
1	12	30	28
2	36	60	104
3	84	120	350
4	144	240	1074
5	120	420	3000
6	0	780	7516

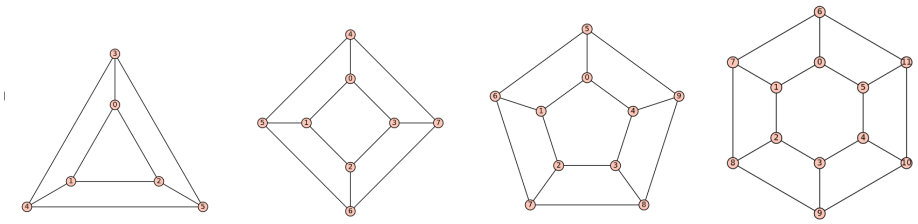
Število induciranih kopij P_k			
k	oktaeder	dodekaeder	ikozaedrski
1	12	27	28
2	12	51	68
3	0	96	102
4	0	135	126
5	0	210	44
6	0	312	2





3.4 Circular Ladder Graphs (CLG)

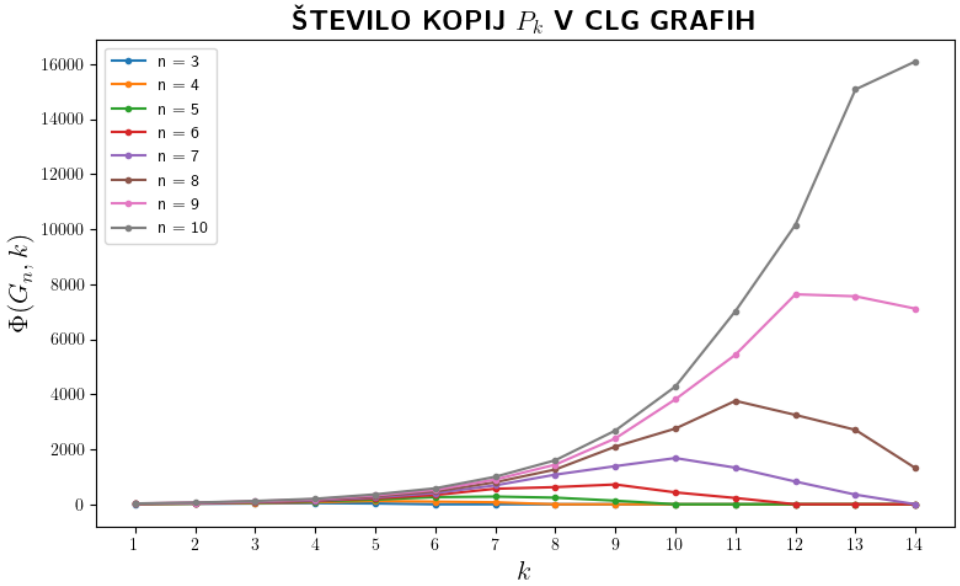
CLG grafe sva prav tako generirala v Sage z ugazom CircularLadderGraph(n). Na spondjih slikah so prikazani CircularLadderGraph(n) za $n = 3, 4, 5, 6$

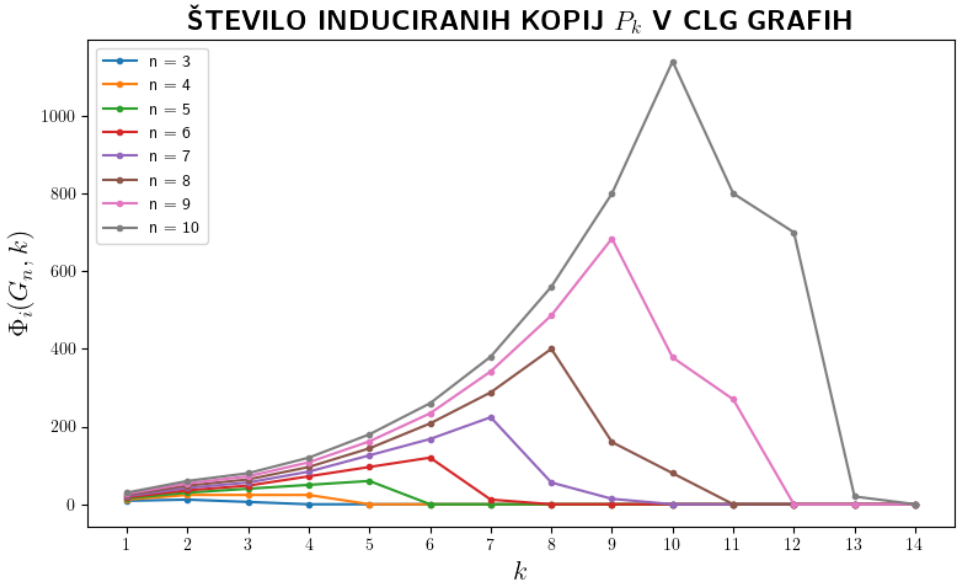


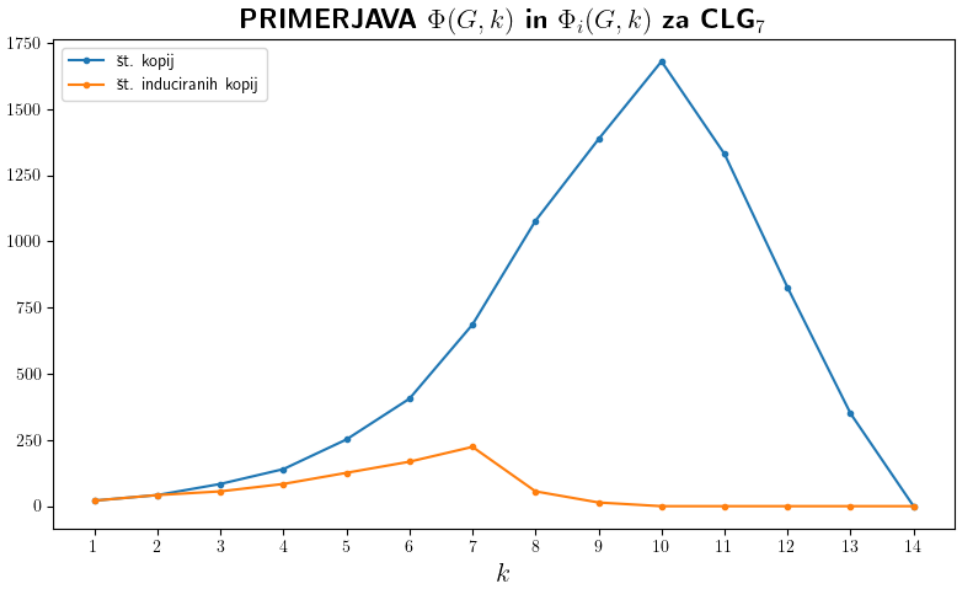
Ta družina se je v smislu izračunljivosti izkazala za daleč najbolj učinkovito. Kot je vidno na grafih sva pri večini grafov iz družine CLG prišla do maksimuma, pri induciranih pa celo čez celotno zalogo vrednosti funkcije ϕ_i. Zadnji graf predstavlja primerjavo med $\phi(G,k)$ in $\phi_i(G,k)$, spet opazimo, da funkcija $\phi(G,k)$ hitro preraste $\phi_i(G,k)$ in maksimum doseže kasneje. Spodnji tabeli prikazujeta izračune funkcij $\phi(G,k)$ (prva tabela) in $\phi_i(G,k)$ (druga tabela) za CLG, za $k = 1, \ldots, 14$, kjer je k dolžina poti.

Število kopij P_k								
k	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
1	9	12	15	18	21	24	27	30
2	18	24	30	36	42	48	54	60
3	30	48	60	72	84	96	108	120
4	42	72	100	120	140	160	180	200
5	30	120	170	216	252	288	324	360
6	0	96	260	336	406	464	522	580
7	0	72	280	564	686	800	900	1000
8	0	0	240	624	1078	1264	1440	1600
9	0	0	130	720	1386	2096	2394	2680
10	0	0	0	432	1680	2752	3816	4280
11	0	0	0	228	1330	3760	5436	7020
12	0	0	0	0	826	3248	7632	10160
13	0	0	0	0	350	2704	7560	15080
14	0	0	0	0	0	1312	7110	16100

Število induciranih kopij P_k								
k	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
1	9	12	15	18	21	24	27	30
2	12	24	30	36	42	48	54	60
3	6	24	40	48	56	64	72	80
4	0	24	50	72	84	96	108	120
5	0	0	60	96	126	144	162	180
6	0	0	0	120	168	208	234	260
7	0	0	0	12	224	288	342	380
8	0	0	0	0	56	400	486	560
9	0	0	0	0	14	160	684	800
10	0	0	0	0	0	80	378	1140
11	0	0	0	0	0	0	270	800
12	0	0	0	0	0	0	0	700
13	0	0	0	0	0	0	0	20
14	0	0	0	0	0	0	0	0




ŠTEVILO INDUCIRANIH KOPIJ P_k V CLG GRAFIH
n = 3
n = 4
n = 5
n = 6
n = 7
n = 8
n = 9
n = 10
$\Phi_i(G_n, k)$
k


PRIMERJAVA $\Phi(G, k)$ in $\Phi_i(G, k)$ za CLG$_7$
št. kopij
št. induciranih kopij
k