Univerza v Ljubljani
Fakulteta za matematiko in fiziko

Bor Rotar, Jani Metež

O ekstremnih grafih v povezavi z obteženim Szegedovim indeksom

Ljubljana, 2020

1 Definiranje problema

V projektni nalogi bova pokazala, da so grafi, ki minimizirajo obteženi Szgadov indeks (wSz), za 26 ali več vozlišč drevesa in jih pokazala. Hkrati bova skušala ugotoviti čim več lastnosti teh dreves.

Obteženi Szgedov indeks:

$$wSz(G) = \sum_{e=uv \in E(G)} [deg(u) + deg(v)] \cdot n_u(e) \cdot n_v(e)$$

.

Pripombe:

1. $deg(u)$ je stopnja vozlišča
2. $n_u(e)$ je moč množice vseh vozlišč, ki so bližje u kot pa v (vključno z u in v)
3. Obteženi Szgadov indeks je definiran za enostavne grafe

Potek dela:

V *Sage*-u oz. *Cocalc*-u bo potrebno definirati wSz in ugotoviti čim bolj enostaven način za generiranje grafov, ki minimizirajo ta indeks. Ko bo to storjeno, bo potrebno le še opaziti čim več možnih lastnosti teh grafov in od katerega števila vozlišč naprej veljajo. Nekatere lastnosti že poznamo iz vira na katergea se navezuje projekt.

2 Algoritmi

2.1 Obteženi Szgadov indeks

Obteženi Szgadov indeks bomo označili z wSz. Definiran bo tako, da se bo zapeljal čez vsako povezavo.

```
def wSz(M):
    indeks = []
    d = M.distance_all_pairs()
    for u,v in M.edges(labels = false):
        blizu_u = 0
        for a in M.vertices():
            if d[a][u] < d[a][v]:
                blizu_u += 1
            blizu_v = order(M)- blizu_u
        indeks += [(M.degree(u) + M.degree(v)) * blizu_u * blizu_v]
    return sum(indeks)
```

2.2 Spreminjanje grafa

Algoritem vzame nek povezan graf in mu dodaja ali odstranjuje povezave, pri tem pazi na to, da graf ostaja povezan (wSz je definiran za enostavne torej povezane grafe).

```
from sage.graphs.connectivity import is_connected
def spremeni_graf(G):
    H = Graph(G)
    if random() < 0.5:
        i = 0
        while True:
            H.delete_edge(H.random_edge())
            if is_connected(H):
                H
                break
            else:
                H = Graph(G)
                i = i + 1
                True
            if i > 15:
                H.add_edge(H.complement().random_edge())
                break
    return H
```

Dodamo še preprosto funkcijo, ki grafu doda novo vozlišče in ga poveže z prvim vozliščem grafa.

```
def novo_vozlisce(G):
    H = Graph(G)
    novo_vzl = order(H)
    H.add_edge((0,novo_vzl,None))
    return H
```

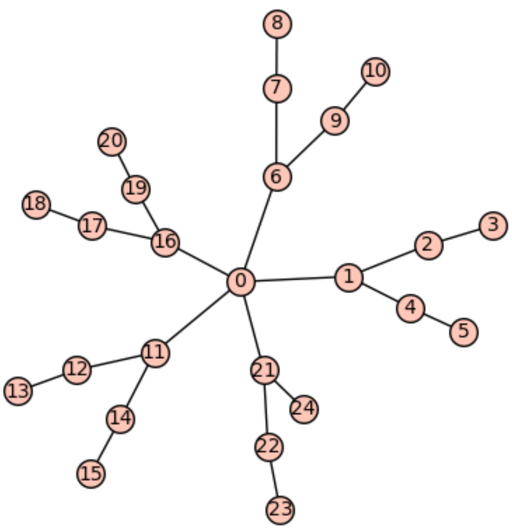
2.3 Algoritem za minimiziranje wSz

Ko pogledamo grafe z minimalnim wSz do 25 vozlišč opazimo, da so si podobni. Sklepamo: če grafu na n vozliščih, ki že ima minimalni wSz, dodamo novo vozlišče s povezavo, bomo lažje prišli do grafa z minimalnim wSz, kot pa če to iščemo na čisto novem grafu.

Torej, da najdemo graf na n vozliščih z $min(wSz)$ bomo grafu na $(n-1)$ vozliščih z že minimiziranim wSz dodali vozlišle in povezavo in ga spreminjali ter te spremembe obdržali, če je wSz novega grafa manjši. Ko wSz ne bomo mogli več zmanjšati, algoritem oz. ponovitve algoritma zaključimo.

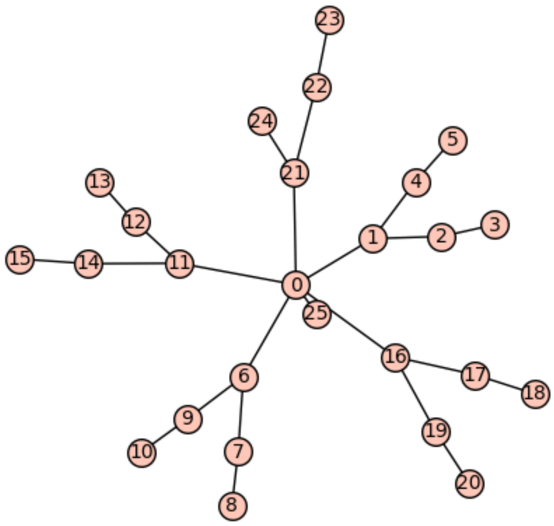
```
def min_wSz(H,koraki):
    k = 0
    primerjajH = H
    HwSz = wSz(H)
    while k < koraki:
        k = k + 1
        H = spremeni_graf(H)
        MwSz = wSz(H)
        if MwSz < HwSz:
            primerjajH = H
    return primerjajH
```

V algoritem pa lahko tudi vstavimo graf za katerega sklepamo, da ima minimalni wSz, in če ne najde boljšega grafa (po zadostnem številu korakov) lahko sklepamo, da ima ta graf že minimalni wSz.

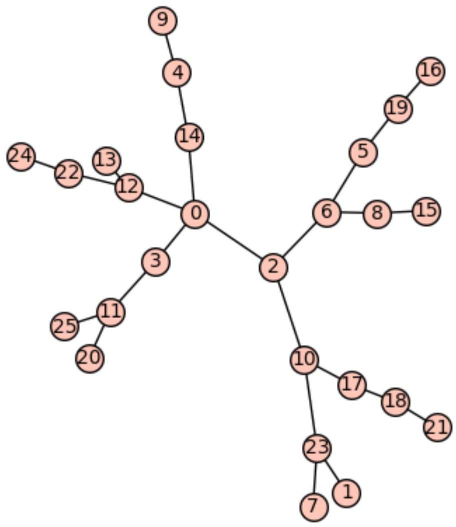


Slika 1: Graf na 25 vozliščih $wSz = 6686$

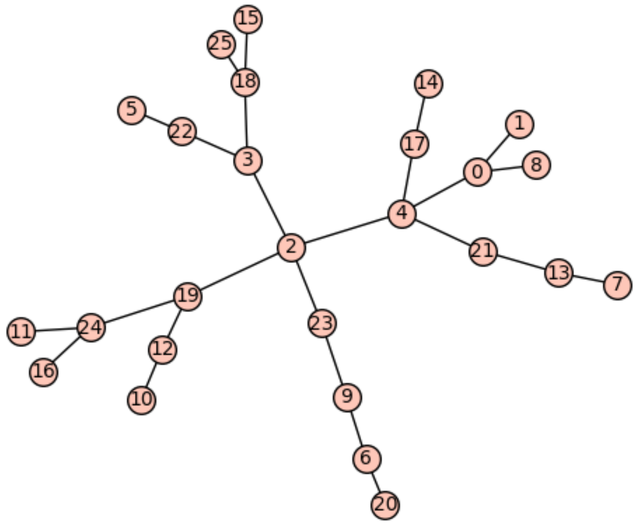
3 Primer iskanja in ugotovitve



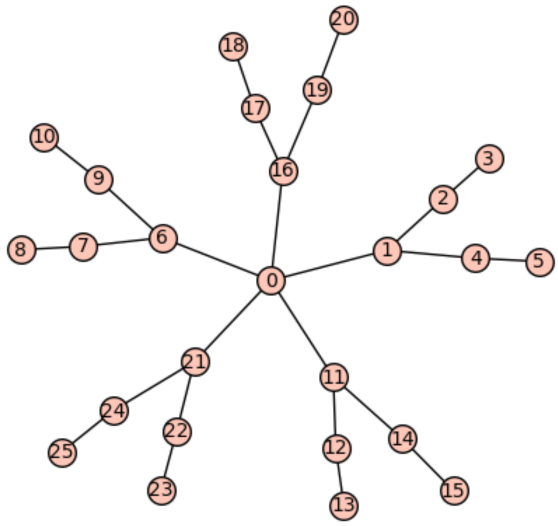
Slika 2: Začetni graf na 26 vozliščih $wSz = 7682$



Slika 3: Graf na 26 vozliščih po 40000 korakih $wSz = 7632$

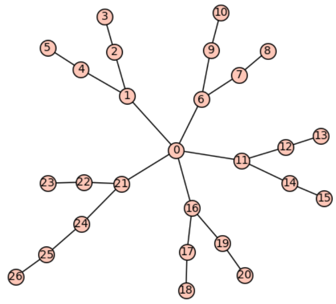


Slika 4: Graf na 26 vozliščih po 80000 korakih $wSz = 7560$

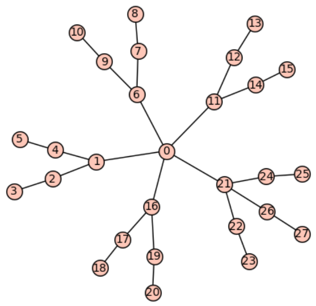


Slika 5: Graf na 26 vozliščih na koncu $wSz = 7350$

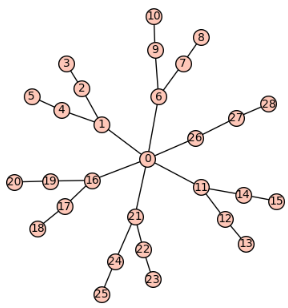
Končni grafi so bili vedno drevesa, brez ciklov in vedno ena povezava manj od števila vozlišč. Rezultati podpirajo predpostavko, da imajo minimalni wSz le drevesa.



Slika 6: Graf na 27 vozliščih na koncu $wSz = 8118$



Slika 7: Graf na 28 vozliščih na koncu $wSz = 8910$



Slika 8: Graf na 29 vozliščih na koncu $wSz = 9864$

Opazimo, kako z večanjem števila vozlišč nastane novo poddrevo, ki je povezano direktno s korenom (vozlišče 0).

4 Lastnosti ekstremnih grafov

Za grafe s 7 do 29 vozlišči smo poiskali wSz in še primerjali s grafom na ena manj vozlišču.

Tabela 1: Izračunani wSz-ji minimalnih grafov in primerjave s prejšnimi

n	wSz	razlika	n	wSz	razlika
7	204	-	19	3292	418
8	306	102	20	3758	466
9	432	126	21	4240	482
10	578	146	22	4806	566
11	762	184	23	5396	590
12	970	208	24	6038	642
13	1210	240	25	6686	648
14	1476	266	26	7350	664
15	1780	304	27	8118	768
16	2100	320	28	8910	792
17	2472	372	29	9864	954
18	2874	402			

5 Viri

Literatura

[1] Jan Boka, Boris Furtulab, Nikola Jedličkova in Riste Škrekovski *On Extremal Graphs of Weighted Szeged Index* https://arxiv.org/abs/1901.04764, 15 Jan 2019.