Metrične dimenzije usmerjenih grafov

Projekt pri predmetu finančni praktikum

Maša Popovič

13. 12. 2024

Opis problema

- Napisati model CLP za določitev metrične dimenzije usmerjenih grafov.
- Pogledati kako se obnaša metrična dimenzija v smeri urinega kazalca usmerjenih cirkulantnih grafov C(n, d). To so močno povezani grafi, torej jim vedno lahko določimo metrično dimenzijo.
- Za mnoge majhne vrednosti n in d, želimo določiti metrično dimenzijo grafa C(n, d) in nato poskusiti uganiti splošno formulo za metrično dimenzijo grafa C(n, d).

Definicije

Definicija 0.1. Naj bo G usmerjen graf z množico vozlišč V in množico povezav E ter $u, v \in V$. Za vozlišče v velja, da je *dosegljivo* iz u, če med u in v obstaja pot. Za dolžino poti med vozliščema vzamemo dolžino najkrajše poti.

Definicija 0.2. Naj bo G graf z množico vozlišč V in množico povezav E ter $u, v, w \in V$. Za vozlišče w pravimo, da *razreši* par vozlišč u in v, če velja, da sta u in v dosegljivi iz w na različnih razdaljah, t.j. $d(w, u) \neq d(w, v)$. Množica vozlišč S razreši graf G, če za vsak par vozlišč v G velja, da ga razreši nek element S. *Metrična dimenzija* grafa G je kardinalnost najmanjše množice, ki zadošča prejšnji lastnosti. To označimo z $\beta(G)$.

Opomba. Množica S, ki razreši dani graf G ni nujno enolična. Preprost primer je npr. graf, ki je pot. Potem je metrična dimenzija enaka 1, za vozlišče v S pa lahko vzamemo eno izmed robnih vozlišč grafa G.

Definicija 0.3. Naj bo G graf z množico vozlišč V in množico povezav E. Naj bo $W = \{w_1, w_2, \ldots, w_n\}, W \subseteq V$ urejena množica in $v \in V$. Oznaki

$$r(v|W) = (d(w_1, v), d(w_2, v), \ldots, d(w_n, v))$$

pravimo metrična reprezentacija v glede na množico W. Sledi, da W razreši G natanko tedaj, ko velja $r(u|W) \neq r(v|W)$ za vse $u, v \in V$, kjer velja $u \neq v$.

Definicija 0.4. *V smeri urinega kazalca usmerjen cirkulantni graf* $C(n, d)$ je usmerjen graf z n vozlišči, kjer so vozlišča označena kot $v_0, v_1, \ldots, v_{n-1}$. Množica povezav E vsebuje usmerjene povezave med vozlišči, in sicer vsakemu vozlišču v_i pripadajo povezave do naslednjih d vozlišč v smeri urinega kazalca. Množica povezav je torej podana kot:

$$E = \left\{(v_i, v_{(i+k) \mod n}) : 1 \leq k \leq d\right\}, \quad \text{za vsak } i \in \{0, 1, \ldots, n-1\}.$$

Celoštevilski linearni program

Koda za iskanje metrične dimenzije usmerjenega grafa, ter minimalne razrešljive množice:

```
def metricna_dimenzija_usmerjenega_grafa(graf):
    if not isinstance(graf, DiGraph):
        return "Napaka: Podani graf ni usmerjen graf."
    V = graf.vertices()

    razdalje = {u: {v: graf.distance(u, v) for v in V} for u in V}

    lp = MixedIntegerLinearProgram(maximization=False)
    x = lp.new_variable(binary=True)
    lp.set_objective(sum(x[v] for v in V))

    for u in V:
        for v in V:
            if u != v:
                vozlisca = [
                    w for w in V
                    if razdalje[w][u] != razdalje[w][v] and
                    razdalje[w][u] < infinity and
                    razdalje[w][v] < infinity
                ]
                if not vozlisca:
                    return f"Metrične dimenzije ni mogoče določiti"

                lp.add_constraint(sum(x[w] for w in vozlisca) >= 1)

    lp.solve()

    razresljiva_mnozica = [v for v in V if lp.get_values(x[v]) == 1]
    return razresljiva_mnozica, len(razresljiva_mnozica)
```

Opis programa

Ta program je implementiran v okolju SageMath in izračuna metrično dimenzijo usmerjenega grafa ter poišče eno razrešljivo množico. Program uporablja metodo linearnega programiranja.

1. Preverjanje vrste grafa:
Prvi korak programa je preverjanje, ali je podani graf usmerjen. Če graf ni usmerjen, se izpiše napaka in program se ustavi.

2. Iskanje razdalj:
Algoritem poišče vse razdalje med poljubnima dvema vozliščema.

3. Ustvarjanje linearnega programa:
Za izračun metrične dimenzije ustvarimo celoštevilski linearni program, kjer so spremenljivke $x[v]$ binarne, kar pomeni, da lahko vsako vozlišče bodisi izberemo bodisi ne (vrednosti 0 ali 1). Cilj programa je minimizirati vsoto vseh teh spremenljivk, kar pomeni, da želimo najti čim manjše število vozlišč, ki bodo zadovoljila vse pogoje.

4. Dodajanje omejitev:
Za vsak par vozlišč u in v v grafu preverimo, ali obstaja vozlišče w, iz katerega razdalji do u in v nista enaki, vendar sta končni. Če takšno vozlišče w obstaja, dodamo omejitev, ki zagotavlja, da bo vsaj eno od teh vozlišč izbrano:

$$\sum_{w \in W(u,v)} x[w] \geq 1$$

Če pa takšno vozlišče za določeni par vozlišč ne obstaja, program vrne napako, ki pove, da metrične dimenzije ni mogoče določiti.

4. Rezultat:
Ko so vse omejitve dodane, program izvede optimizacijo in poišče rešitev linearnega programa. Cilj je, da minimiziramo množico, ki razreši graf. Program to množico tudi vrne, poleg tega pa vrne tudi njeno kardinalnost tj. metrično dimenzijo grafa.

Časovna zahtevnost programa

Naj bo V število vozlišč v grafu, E pa število povezav.

- **Preverjanje usmerjenosti grafa:** $O(1)$ - enostavno preverjanje vrste grafa.
- **Seznam vozlišč:** $O(V)$
- **Izračun razdalj med vsemi pari vozlišč:** $O(V^2+V \cdot E)$ - pri usmerjenem grafu je razdalja (u, v) običajno izračunana z algoritmom BFS. Za izračun razdalj od enega vozlišča u do vseh ostalih vozlišč v je časovna zahtevnost $O(V + E)$. To izvajamo za vsak $u \in V$, kar daje: $O(V \cdot (V + E))$
- **Omejitve:** $O(V^3)$ - tukaj imamo dve zanki ki iterirata po vseh parih (u, v), kar vzame $O(V^2)$ časa. Vsak pogoj preveri v konstantnem času, torej skupaj $O(V)$.

- **Reševanje linearnega programa:** $O(2^V)$ - reševanje linearnega programa je NP-težak problem, časovna zahtevnost je približno $O(2^V)$ v najslabšem primeru.
- **Pridobivanje rezultatov:** $O(V)$

Skupna časovna zahtevnost programa je torej:

$$O(V^3 + V \cdot E + 2^V)$$

Časovna zahtevnost bo postala problematična pri večjih grafih, predvsem zaradi iskanja razdalj in reševanja linearnega programa.

Generiranje podatkov

Najprej sem kodo uporabila na nekaj preprostih usmerjenih grafih. Te grafe sem generirala s funkcijo `DiGraph`.

Uporaba CLP na cirkulantnih grafih

V drugem delu projekta sem opazovala metrične dimenzije usmerjenih cirkulantnih grafov $C(n, d)$. Posamezne cirkulantne grafe sem generirala z naslednjo kodo:

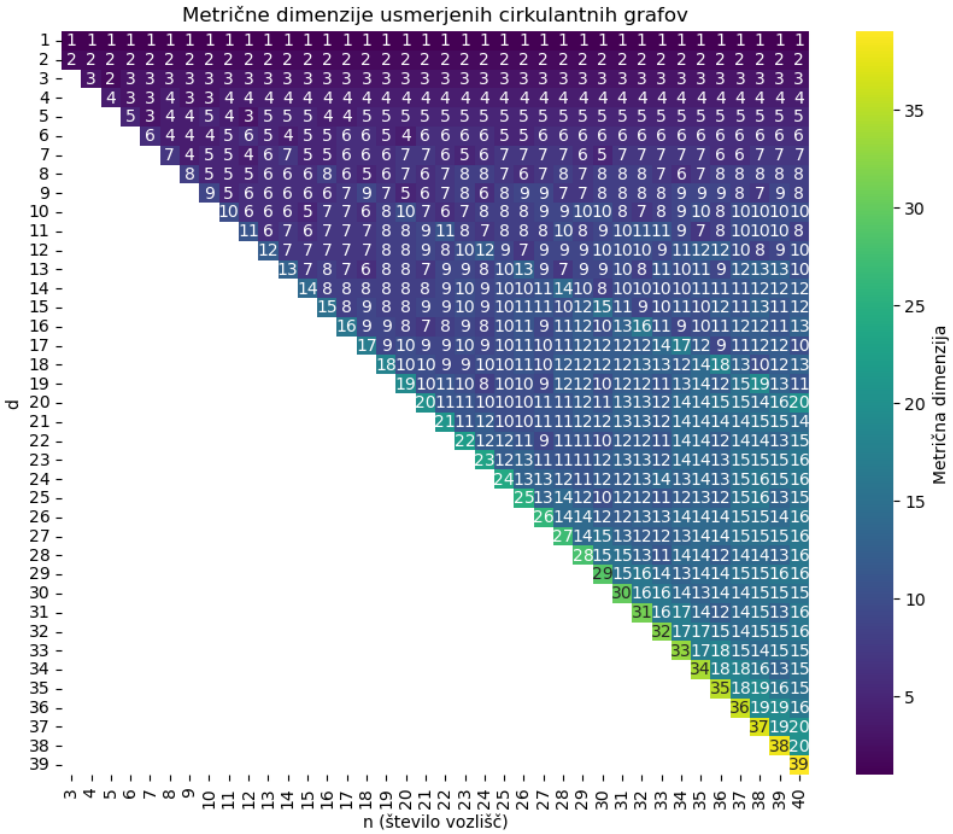
```
def clockwise_circulant_graph(n, d):
    odmiki = list(range(1, d + 1))
    return digraphs.Circulant(n, odmiki)
```

Nato pa sem izračunala metrične dimenzije cirkulantnih grafov $C(n, d)$ za vse $n \leq 40$ in $d < n$. Uporabila sem naslednjo kodo:

```
def generiraj_podatke_za_circulant_graphs_od_do(min_n, max_n):
    podatki = []
    for n in range(min_n, max_n + 1):
        for d in range(1, n):
            G = clockwise_circulant_graph(n, d)
            _, dimenzija = metricna_dimenzija_usmerjenega_grafa(G)
            podatki.append((n, d, dimenzija))
    return podatki
```

Koda deluje dobro za majhne grafe, za večje grafe ($n > 30$) pa koda dela precej počasneje. To je seveda zaradi linearnega programa, ki za večje grafe deluje zelo počasi.

Izračunane metrične dimenzije sem shranila v csv dokument, za enostavnejšo kasnejšo analizo. Rezultate sem prikazala tudi na grafu.



Slika 1: Metrične dimenzije

Ugotovitve

Za poljuben n velja:

$$\beta(C(n,1)) = 1$$
$$\beta(C(n,2)) = 2$$
$$\beta(C(n,n-1)) = n-1$$

Za majhne n, in sicer $n < 12$ velja:

$$\beta(C(n,d)) = \left\lfloor \frac{n}{2} \right\rfloor, \quad \text{za} \quad \left\lceil \frac{n}{2} \right\rceil \leq d < n-1$$

Torej bi lahko za majhne n rekli, da se metrična dimenzija stabilizira, in da je točka stabilizacije $d = \left\lceil \frac{n}{2} \right\rceil$. Pravilo ni čisto veljavno, saj imamo en proti primer in sicer $\beta(C(10,6)) = 4$. Po tem pravilu pa bi moglo biti $\beta(C(10,6)) = 5$.

Lahko opazimo, da za fiksen n metrične dimenzije naraščajo približno linearno, ko večamo d (za $d < n-1$). Pri fiksnem n je maksimalna metrična dimenzija dosežena pri $d = n-1$, kjer ima vrednost:

$$\beta(C(n,n-1)) = n-1$$

Za fiksen n in $d < n-1$ pa je maksimalna metrična dimenzija dosežena ko je $d = n-2$ in $d = n/2$, kjer ima vrednost:

$$\beta(C(n,n-2)) = \beta\left(C\left(n,\frac{n}{2}\right)\right) = \left\lfloor \frac{n}{2} \right\rfloor$$

Za fiksen $n > 4$, je ta maksimum dosežen tudi v $d = n - 3$. Torej:

$$\beta(C(n, n-3)) = \left\lfloor \frac{n}{2} \right\rfloor \quad \text{za} \quad n > 4$$

Na podlagi ugotovitev, lahko zapišemo **zgornjo mejo**. In sicer, za poljbna n in d velja:

$$\beta(C(n,d)) \leq n - 1 \quad \text{oz.} \quad \beta(C(n,d)) \leq d$$

Za poljuben n in $d < n - 1$ pa velja:

$$\beta(C(n,d)) \leq \left\lfloor \frac{n}{2} \right\rfloor$$

Boljša zgornja meja:
Lahko opazimo, da se metrična dimenzija obnaša podobno kot logaritemska funkcija, ko večamo d. Za $\lceil n/2 \rceil < d < n - 1$ je naslednja zgornja meja bolj natančna:

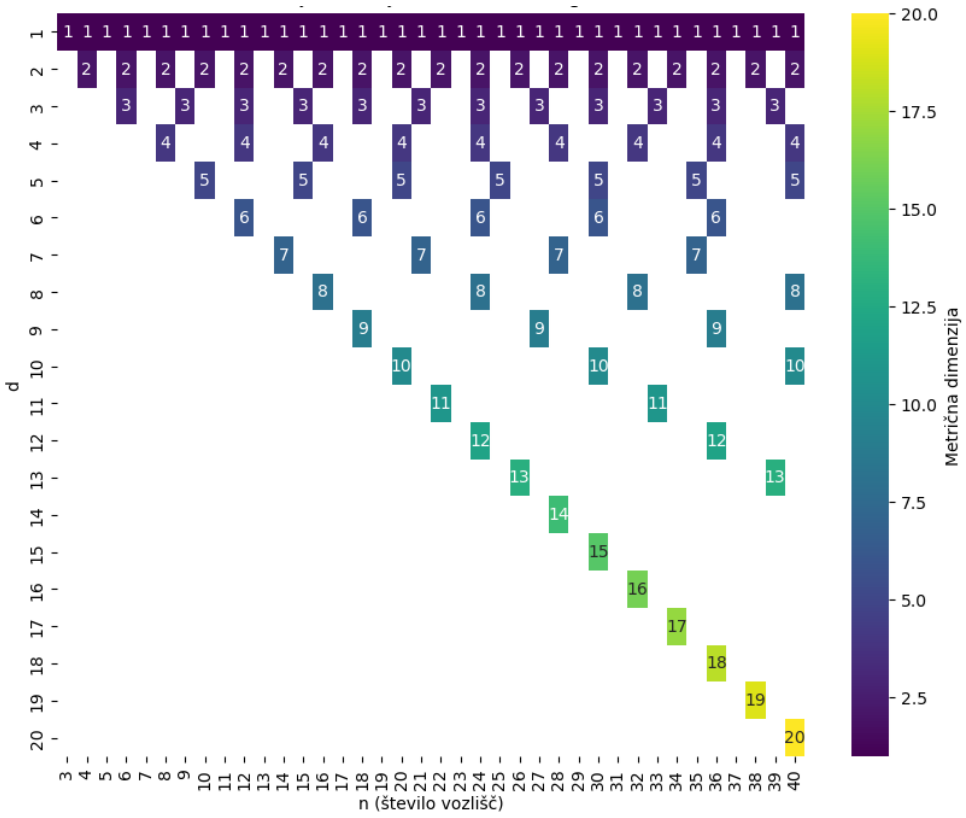
$$\lfloor \sqrt{d} * \ln(d+1) \rfloor + 1$$

Torej

$$\text{zgornja meja} = \begin{cases} d & \text{če } d \leq \lfloor n/2 \rfloor, \\ \lfloor \sqrt{d} * \ln(d+1) \rfloor + 1 & \text{če } \lceil n/2 \rceil < d < n - 1. \end{cases}$$

Do tega sem prišla na podlagi opazovanja cirkulantnih grafov za $n \leqslant 40$. Opazila pa sem dva grafa za katera ta meja ne velja. In sicer $C(36, 20)$ in $C(37, 20)$. Seveda to lahko hitro rešimo, če zgornjo mejo povečamo za ena, vendar izgubimo na natančnosti pri ostalih grafih.

Cirkulantni grafi oblike $\mathbf{C(c \cdot d, d)}$:



Slika 2: Metrične dimenzije usmerjenih cirkulantnih grafov za $n = cd$

Opazimo, da za vse $n = c \cdot d$, kjer je c neka konstanta, velja:

$$\beta(C(cd, d)) = d$$

Torej pri fiksnem d velja, da je metrična dimenzija enaka d v vseh tistih n, ki so večkratniki števila d. Torej na primer pri $d = 10$ zavzame maksimume (tj. vrednosti $\beta = 10$) v $n \in \{20, 30, 40 \dots\}$. To pa ne pomeni, da so to edini n pri fiksnem d, kjer metrična dimenzija doseže vrednost d.