

Graffiti conjecture 299

Poročilo projekta

Projekt v povezavi s predmetom Operacijske raziskave

Avtorja:

Stefan Đekanović, Katja Marina

Ljubljana, november 2018

Kazalo

1	Opis projekta	2
2	Definicije	2
3	Načrt dela	3
4	Primer	3
5	Pomožne funkcije	4
5.1	degrees(G)	4
5.2	powerset_S	4
5.3	annihilation(G))	4
5.4	total_domination_number(G))	5
6	Glavna funkcija	5
7	Iskanje protiprimera	5

1 Opis projekta

Graffiti je računalniški program, ki programira grafsko-teoretične domneve. Program pozna določene grafe in je sposoben ovrednotiti določene formule oblikovane iz teoretičnih invariant grafov. Če nobeden od grafov, s katerimi je seznanjen Graffiti, ni protiprimer za formulo, to formulo imenujemo domneva. Glavni problem je število teh domnev, še posebej tistih, ki so trivialne.

Najin projekt je testiranje domneve 299, ki pravi:

Če je G enostaven povezan graf, potem je

$$g_t(G) \leq \text{Annihilation}(G) + |S|,$$

kjer je S množica vozlišč, ki imajo stopnjo 2.

2 Definicije

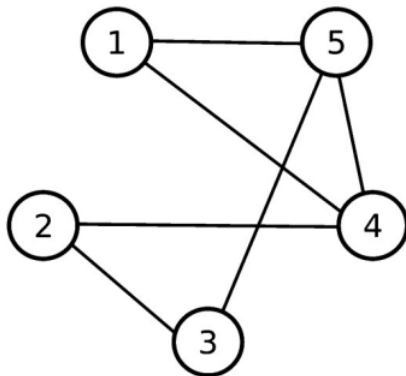
- Graf G je *povezan*, če za poljubni vozlišči $u, v \in G$ obstaja pot od u do v .
- Graf G je *enostaven*, če nima zank (začetna točka ni tudi njena končna točka) in vzporednih povezav (povezavi, ki imata skupno začetno in končno točko).
- *Dominantna množica* oziroma *dominating set* je podmnožica vozlišč grafa G , za katero velja, da dominira vse ostale točke v grafu. Točka v grafu dominira drugo točko, če sta enaki, ali pa sta med seboj povezani (sosedni).
- *Totalna dominantna množica* oziroma *total dominating set* $D(G)$ grafa G je dominantna množica vozlišč grafa G , v kateri ima vsako vozlišče iz dominantne množice tudi vsaj enega soseda iz te množice.
- *Moč totalne dominantne množice* označimo z $g_t(G)$.
- *Totalno dominantno število grafa* G je moč najmanjše totalne dominantne množice grafa.
- *Stopnja točke* v neusmerjenem grafu G je število vseh njenih sosednih točk.
- *Annihilation*(G) oziroma »uničenje« grafa G – $A(G)$ definiramo tako: Naj ima graf G n vozlišč in naj bo d_1, d_2, \dots, d_n nepadajoče zaporedje stopenj grafa G . $A(G)$ je največje celo število k , da je vsota prvih k členov zaporedja največ $\frac{(d_1 + d_2 + \dots + d_n)}{2}$.

3 Načrt dela

Delo sva začela s pisanjem pomožnih funkcij, ki so nama v nadaljevanju pomagale pri glavni funkciji za testiranje hipoteze. Funkcija na koncu vrne, ali je bila domneva ovržena, ali pa jo potrdi. Vendar pa ta program učinkovito deluje le za (približno) $n \in \{1, 2, \dots, 8\}$ zaradi časovne zahtevnosti, zato sva se lotila iskanja protiprimera z eno od populacijskih metahevrstičnih metod, in sicer z genetskim algoritmom.

4 Primer

Prikazala bova primer testiranja domneve na manjšem enostavnem povezanem grafu s 5 vozlišči.



$$S = \{1, 2, 3\} \Rightarrow |S| = 3$$

$$g_t(G) = |\{3, 5\}| = 2$$

$$d_1 \leq d_2 \leq d_3 < d_4 < d_5 :$$

$$d_1 = 2, d_2 = 2, d_3 = 2, d_4 = 3, d_5 = 3$$

$$\frac{d_1 + d_2 + d_3 + \dots + d_5}{2} = \frac{2 + 2 + 2 + 3 + 3}{2} = 6$$

$$d_1 + d_2 + d_3 = 6 \Rightarrow k = 3, \text{ torej je } A(G) = 3.$$

Najina domneva pravi:

$$2 \leq 3 + 3 = 6, \text{ torej v tem primeru drži.}$$

5 Pomožne funkcije

Najina enačba je sestavljena iz treh spremenljivk, ki jih bova izračunala s pomočjo pomožnih funkcij.

5.1 degrees(G)

Najprej sva definirala funkcijo, ki vrne seznam vozlišč grafa, s katero si bova pomagala pri računanju *annihilation(G)*.

```
def degrees(G):
    stopnje = G.degree_sequence()
    return stopnje
```

5.2 powerset_S

Druga pomožna funkcija se imenuje *powerset_S*, ki iz seznama stopenj vozlišč vrne število vozlišč stopenj 2.

```
def powerset_S(G):
    velikost_S = degrees(G).count(2)
    return velikost_S
```

5.3 annihilation(G)

Funkcija *annihilation(G)* najprej vzame seznam stopenj vozlišč, in ga obrne, tako da so v naraščajočem zaporedju. Potem izračuna polovično vsoto vseh stopenj, kar rabimo v nadeljevanju funkcije. Z zanko for se sprehodimo po naraščajočem seznamu stopenj, in ko je vsota i-tih stopenj še manjša ali pa enaka polovični vsoti stopenj, vrne število i.

```
def annihilation(G):
    stopnje = degrees(G)
    stopnje_obr = stopnje[::-1]
    vel = int(sum(stopnje_obr)/2)
    sest = 0
    if stopnje_obr == [0]:
        return 1
    else:
        for i, k in enumerate(stopnje_obr):
            sest += k
            if sest > vel:
                return i
```

5.4 total_domination_number(G))

Naslednja pomožna funkcija je *total_domination_number(G)*, ki z mešanim celoštevilskim linearnim programom izračuna totalno dominantno število grafa. Pri tem sva ločila še primer, ko ima graf samo eno vozlišče in ima totalno dominantno število 0, in pa primer, ko sta vozlišči dve, kjer je totalno dominantno število 1.

```
def total_domination_number(G):
    if len(G.vertices()) == 1:
        return 0
    p = MixedIntegerLinearProgram(maximization = False)
    b = p.new_variable(binary = True)
    p.set_objective( sum([b[v] for v in G]) )

    for u in G:
        p.add_constraint( sum([b[v] for v in G[u]]) >= 1 )

    p.solve()
    b = p.get_values(b)
    return len([v for v,i in b.items() if i])
```

6 Glavna funkcija

Vse pomožne funkcije, ki sva jih definirala, na koncu združiva v funkciji *testiranje_domneve*, ki deluje za majhne grafe, do približno $n = 8$. Če za n vstaviva večje število, najin algoritem deluje počasneje zaradi velike časovne zahtevnosti.

```
def testiranje_domneve(n):
    grafi = list(graphs.nauty_geng(str(n)+" -c"))
    for graf in grafi:
        if total_domination_number(graf) > (annihilation(graf))
+ (powerset_S(graf)):
        show(graf)
        print(annihilation(graf))
        return ("Domneva je ovrzena.")
    return "Domneva ni ovrzena."
```

7 Iskanje protiprimer

Glavna funkcija v nobenem primeru ne ovrže, zato se morava problema lotiti še drugače, to pa je z uporabo metahevrstike, bolj natančno populacijske metode. V njej bova uporabila tudi pomožne funkcije od prej. Zaradi zahtevnosti pa ni zagotovljeno, da bova protiprimer našla.