Graphs of type (SB) and domination on their cartesian products

Jan Hrastnik, Matic Kremžar

11. 12. 2024

1 Uvod

V projektni nalogi sva se ukvarjala z grafi tipa (SB) in dominacijo na njihovem kartezičnem produktu. Projektna naloga je bila izvedena v programu SageMath, obsežni izračuni pa s pomočjo spletne strani CoCalc. Naloga ima odprt tudi svoj repozitorij na GitHubu, na naslovu https://github.com/matickremzar/Graphs-of-type-SB-and-domination-on-their-cartesian-products-/tree/main. Tam so zbrane tudi vse datoteke, ki sva jih uporabila med izdelavo projekta.

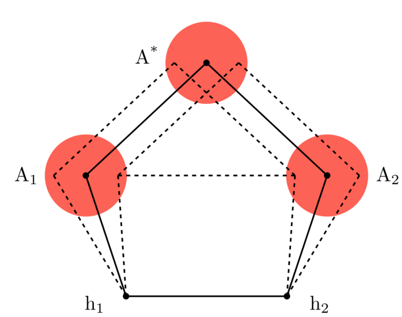
Cilj je razumeti, kako prepoznati, konstruirati in spreminjati grafe tipa (SB), ter analizirati njihove lastnosti, zlasti dominacijsko število njihovih kartezičnih produktov.

2 Osnovna ideja in definicije

Delala sva z usmerjenimi grafi $G = (V, E)$, kjer je G množica vozlišč, E pa množica povezav grafa.

Definicija 1. *Graf G je tipa (SB), če je njegov premer enak 2 in ima dve sosednji vozlišči $v_1, v_2 \in V$, da velja:*

- *v_1 in v_2 nimata skupnega soseda,*
- *G ima vozlišče $v^* \in V$, ki ni sosednje v_1 ali v_2; torej $v^* \not\sim v_1$ in $v^* \not\sim v_2$.*



Slika 1: Skica grafov tipa (SB)

Zapišemo lahko particijo vozlišč grafa G kot

$$V(G) = v_1, v_2 \cup A_1 \cup A_2 \cup A^*,$$

kjer je A_1 množica vozlišč, ki so sosednja v_1, A_2 množica vozlišč, ki so sosednja v_2 in A^* množica vozlišč, ki niso sosednja niti v_1 niti v_2.

Definicija 2. *Podmnožica vozlišč $D \subseteq V$ grafa $G = (V, E)$ se imenuje* dominacijska množica *grafa, če je vsako vozlišče $v \in V$ grafa v množici D, ali pa je kakšno njemu sosednje vozlišče v D.* Dominacijsko število $\gamma(G)$ *grafa je velikost najmanjše dominacijske množice grafa.*

Definicija 3. *Kartezični produkt $G \square H$ grafov G in H je graf, za katerega velja:*

- *vozlišča grafa $G \square H$ so kartezični produkt $V(G) \times V(H)$,*
- *dve vozlišči (u, v) in (u', v') sta sosednji v grafu $G \square H$ natanko tedaj, ko je:*
  - *$u = u'$ in v je sosed v' v H,* ***ali***
  - *$v = v'$ in u je sosed u' v G.*

3 Naloga 1

3.1 Preverjanje, ali je graf tipa (SB)

Sprva je potrebno zapisati kodo, ki preveri, ali je dan graf tipa (SB) (v datoteki *sb_detect_koncna.py*). Uporabimo matriko sosednosti. Dovolj je, da pregledamo zgolj zgornji trikotnik matrike, saj tako že pridobimo vse podatke o sosedih. Nato preverimo še veljavnost prej zapisanih lastnosti grafov tipa (SB).

3.2 Iskanje vseh grafov tipa (SB) na do 10 vozliščih

Najprej sva v datoteko *diam_two_graphs.txt* zbrala vse grafe s premerom 2 na do 10 vozliščih. Nato sva te grafe prefiltrirala s spodnjo kodo in podatke zbrala v novi datoteki *sb_graphs.txt*, ki je dostopna preko hiperpovezave v README datoteki na GitHub repozitoriju projekta. Datoteka je velika privližno 400 MB.

Z zadnjo kodo še preverimo, koliko je grafov tipa (SB) za posamezen n (število vozlišč), manjši ali enak 10. Rezultati so prikazani spodaj.

n (število vozlišč)	Število grafov tipa (SB)
0	0
1	0
2	0
3	0
4	0
5	2
6	11
7	116
8	1688
9	43420
10	2079097

Tabela 1: Tabela števila grafov tipa (SB) glede na število vozlišč

4 Naloga 2

Namen te naloge je naključno konstruirati grafe tipa (SB) za večje n. Vzamemo nek graf na n vozliščih in ga s pomočjo smiselnega odvzemanja in dodajanja povezav (poskušamo doseči na particijo vozlišč, da pridobimo pogoje za (SB)) poskušamo spremeniti v graf tipa (SB). Pri tem moramo paziti na premer.

Algorithm 1 Preverjanje, ali je graf tipa (SB)

```
function IsSB(G: Graph)                                    ▷ Funkcija za preverjanje grafa tipa (SB)
    if G.diameter() ≠ 2 then return False                  ▷ Premer mora biti 2
    adj ← G.adjacency_matrix()                             ▷ Pridobimo matriko sosednosti
    for i ← 1 to adj.nrows() do
        for j ← i + 1 to adj.nrows() do                    ▷ Samo zgornji trikotnik
            if adj[i, j] = 1 then                          ▷ Če sta i in j soseda
                common_neighbour ← False
                for k ← 1 to adj.nrows() do
                    if adj[i, k] = 1 and adj[j, k] = 1 then    ▷ Preverimo skupne sosede
                        common_neighbour ← True
                        break
                    end if
                end for
                if common_neighbour then
                    continue
                end if
                nonadj_vertex_exists ← False
                for k ← 1 to adj.nrows() do
                    if adj[i, k] = 0 and adj[j, k] = 0 and k ≠ i and k ≠ j then
                        nonadj_vertex_exists ← True
                        break
                    end if
                end for
                if nonadj_vertex_exists then
                    return True
                end if
            end if
        end for
    end for
    return False                                           ▷ Graf ni tipa (SB)
end function
```

Algorithm 2 Filtriranje grafov tipa (SB)

```
Odpri datoteko 'sb_graphs.txt' v načinu dodajanja ("a").
Odpri datoteko "diam_two_graphs.txt" v načinu branja ("r").
for vsako vrstico line v datoteki "diam_two_graphs.txt" do
    g ← pretvori_v_graf(line)          ▷ Ustvari graf iz podatkov na trenutni vrstici.
    if is_sb(g) then                   ▷ Preveri, če je graf g tipa (SB).
        Zapiši line v datoteko 'sb_graphs.txt'.
    end if
end for
Zapri obe datoteki.
```

Algorithm 3 Preštevanje grafov tipa (SB) z različnim številom vozlišč

```
counter ← [0, 0, 0, ..., 0] ▷ (seznam s 11 ničlami, za štetje grafov z različnim številom vozlišč)
Odpri datoteko 'sb_graphs_g6.txt'
for vsako vrstico line v datoteki do
    g ← pretvori_v_graf(line)                                  ▷ (ustvarimo graf iz vrstice)
    counter[len(g.vertices())] ← counter[len(g.vertices())] + 1
end for
Izpis rezultatov:
for vsak i od 0 do 10 do
    izpiši "Število grafov tipa (SB) z i vozlišči: counter[i] "
end for
```

5 Naloga 3

Želimo pridobiti tudi nov graf tipa (SB) iz danega grafa tipa (SB) tako, da naredimo nekaj manjših naključnih modifikacij (npr. dodajanje/odvzemanje vozlišč/povezav).
Koda je dostopna v datoteki *random_modify_SB_graph.py* na GitHubu, zaradi dolžine pa je zapisana le glavna ideja programa.
Funkcija *random_modify_sb_graph* kot argument vzame graf tipa (SB) in ga najprej shrani (kot varovalo za kasneje, če naključne spremembe ne dajo grafa tipa (SB)). Zapiše particijo vozlišč in se odloči za naključno število naključnih sprememb iz nabora. Te spremembe so same po sebi lahke za sprogramirati, paziti je treba zgolj, da ne prihaja do kakšnih neželenih struktur (samozanke ...). Število sprememb je na začetku kolikor toliko majhno, da ne tvegamo, da bi graf preveč spremenili iz tipa (SB). Program izvede te spremembe in preveri ali nov graf ustreza tipu (SB). V kolikor ne, se postopek na takšnem grafu nadaljuje do tisočkrat, kasneje pa spet začnemo z originalnim grafom, ker bi bil novi graf lahko že precej degeneriran.
Funkcijo sva stestirala na približno 5000 primerih, pri čemer je za n blizu 30 postopek v okoli 95% primerov vrnil graf tipa (SB) preden je bilo izvedenih 1000 poskusov, torej preden smo spet vzeli prvotni graf. Skoraj vedno, ko je bilo generiranje grafa uspešno v teh prvih 1000 poskusih, se je to zgodilo v prvih 5 poskusih, kar potrjuje smiselnost zaustavitvenega pogoja z *while* zanko. Sklepava, da so namreč bili grafi potem že zelo 'oddaljeni' od tipa (SB). Za velike n je bil pri teh neuspešnih poskusih verjetno največkrat problem, da se je povečal premer grafa.

6 Naloga 4

Poglejmo za začetek dominacijsko število na običajnih grafih tipa (SB). Intuitivno jasno je, da imajo vsi grafi tipa (SB) dominacijsko število najmanj 2. Tudi dokaz je jasen: Zapišimo particijo vozlišč nekega grafa tipa (SB), kot je predlagano v osnovni ideji problema. Če je v najmanjši dominacijski množici vozlišče v_1, imamo zraven pokrito še v_2 in A_1. Vemo pa, da je A^* neprazna, saj obstaja vozlišče v^*, ki ni sosednje v_1 ali v_2. Torej v_1 ni zadosten in potrebujemo vsaj še eno vozlišče v dominacijski množici. Analogno sledi za v_2. Če je v dominacijski množici neko vozlišče iz A^* ali A_2 (A_1), rabimo še neko vozlišče, ki je sosedno v_1 (v_2).
V praksi se je izkazalo, da ima velika večina grafov tipa (SB) na do 10 vozliščih dominacijsko število enako 2, preostali pa 3.
Nato lahko vpeljemo Vizingovo domnevo: Za grafa G in H velja

$$\gamma(G) * \gamma(H) \leq \gamma(G \square H).$$

Ta sicer ni dokazana v splošnem, je pa preverjena za zadosten nabor grafov, da jo je smiselno uporabiti v tej nalogi. Iz tega lahko trdimo, da je najnižje možno dominacijsko število na kartezičnem produktu dveh grafov tipa (SB) enako 4.
Zanimiva variacija problema bi bila, v kolikor bi bil kartezični produkt nekih dveh grafov tipa (SB)

Algorithm 4 Generiraj naključen SB graf - naloga 2

```
Vhod: Celo število n (število vozlišč)
Izhod: Graf g tipa SB z premerom 2
g ← Naključen graf z n vozlišči in verjetnostjo povezave 0.5
(h1, h2) ← prvi rob grafa g
a1 ← sosedi vozlišča h1
a2 ← sosedi vozlišča h2
Odstrani h2 iz seznama a1
Odstrani h1 iz seznama a2
a_star ← prazen seznam
for vsako vozlišče v v grafu g do
    if v ∉ a1 in v ∉ a2 in v ≠ h1 in v ≠ h2 then
        Dodaj v v seznam a_star
    end if
end for
vertices_to_remove ← prazen seznam
for vsako vozlišče n1 v a1 do
    for vsako vozlišče n2 v a2 do
        if n1 == n2 then
            Odstrani rob (h1, n1)
            Dodaj n1 v seznam vertices_to_remove
        end if
    end for
end for
for vsako vozlišče v v seznamu vertices_to_remove do
    Odstrani v iz seznama a1
end for
if dolžina seznama a1 je 0 then
    u ← dodaj novo vozlišče
    Dodaj u v seznam a1
    Dodaj rob (h1, u)
end if
if dolžina seznama a2 je 0 then
    u ← dodaj novo vozlišče
    Dodaj u v seznam a2
    Dodaj rob (h2, u)
end if
if dolžina seznama a_star je 0 then
    u ← dodaj novo vozlišče
    Dodaj u v seznam a_star
    v_a1 ← prvi element seznama a1
    v_a2 ← prvi element seznama a2
    Dodaj rob (u, v_a1)
    Dodaj rob (u, v_a2)
end if
for vsako vozlišče u v seznamu a1 do
    for vsako vozlišče v v seznamu a_star do
        Dodaj rob (u, v)
    end for
end for
for vsako vozlišče u v seznamu a2 do
    for vsako vozlišče v v seznamu a_star do
        Dodaj rob (u, v)
    end for
end for
diam ← premer grafa g
if diam == 2 in g je tipa SB then
    Vrni graf g
end if
```

spet graf tipa (SB). Posledično bi takoj sledilo, da obstajajo grafi tipa (SB) z dominacijskim številom vsaj 4. Ko sva pregledala grafe na do 10 vozliščih, takšnega primera nisva našla. Problem je bil seveda premer kartezičnih produktov grafov. Vseh 125000 grafov, ki sva jih tako generirala je imelo premer 4.
Iskanje zgornjih mej za dominacijsko število kartezičnega produkta dveh grafov tipa (SB) je zahtevnejša. Najboljša zgornja meja, ki sva jo uspela najti je še ena Vizingova izpeljava:

$$\gamma(G \Box H) \leq min\{\gamma(G)|H|, \gamma(H)|G|\}$$