Finančni praktikum

k-total rainbow domination numer vs. domination number

Tim Resnik
Lana Herman
Univerza v Ljubljani

Fakulteta za matematiko in fiziko

November, 2019

1 Problem naloge

V projektni nalogi se bova ukvarjala z domeno, ki se ukvarja s povezavo med "k-rainbow total domination number"(označimo z $\gamma_{krt}(G)$) in "domination number"(označimo z $\gamma(G)$). Definiciji za $\gamma_{krt}(G)$ in $\gamma(G)$ sta v razdelku **Razlaga pojmov**.
Domneva pravi, da za graf G in $k \geq 4$ obstaja tesna povezava $\gamma_{krt}(G) \geq 2\gamma(G)$. Cilj najine projetkne naloge je najti tak graf, za katerega ta neenkost ne drži. To sva na majhnih grafih preverila na konkretnih primerih, kjer sva število vozlišč in število k vnesla ročno. Za večje grafe sva uporabila metodo *Simulated Annealing*.
Poiskala sva tudi primere, za katere velja enakost $\gamma_{krt}(G) = 2\gamma(G)$.

2 Razlaga pojmov

Graf G ima množico vozlišč $V(G)$ in množico povezav $E(G)$. Za množico $N_G(v)$ velja, da vsebuje vsa sosednja vozlišča v, v grafu G. Za grafa G in H, je kartezični produkt $G \Box H$ graf z množico vozlišč $V(G) \times V(H)$.
Dominirana množica grafa G je $D \subseteq V(G)$, taka da za vsako vozlišče $v \in V(G)$ in $v \notin D$ velja, da je sosed nekemu vozlišču iz D. *Dominirano število*, $\gamma(G)$, je velikost najmanjše dominirane množice. Če za $\forall v \in V(G)$ velja, da je sosed vozlišču iz D, za D rečemo, da je *totalno dominirana množica* grafa G. *Totalno dominirano število*, $\gamma_t(G)$, je velikost najmanjše totalno dominirane množice.
Za pozitivno celo število k, je *"k-rainbow domination function"* (kRDF) grafa G funkcija f, ki slika iz $V(G)$ v množico $\{1, \cdots, k\}$. Zanjo velja, da za katerikoli $v \in V(G)$ in $f(v) = \emptyset$ velja $\cup_{u \in N_G(v)} f(u) = [k]$. Definiramo $\|f\| = \sum_{v \in V(G)} |f(v)|$. $\|f\|$ rečemo *teža* f-a. *"k-rainbow domination number"*, $\gamma_{kr}(G)$, grafa G je minimalna vrednost $\|f\|$ za vse "k-rainbow domination functions". Po definiciji vemo, da za vse $k \geq 1$ velja

$$\gamma_{kr}(G) = \gamma(G \Box K_k).$$

Graf K_k predstavlja polni graf na k vozliščih. Nazadnje definirajmo še *"k-rainbow total domination function"* (kRTDF), katera se od "k-rainbow domination function"razlikuje v dodatnem pogoju, ki zagotavlja, da če za $\forall v \in V(G)$ velja $f(v) = \{i\}$, potem obstaja tak $u \in N_G(v)$, da je $i \in f(u)$. *"k-rainbow total domination number"*, $\gamma_{krt}(G)$, grafa G je minimalna vrednost $\|f\|$ za vse "k-rainbow total domination functions". Tudi tu za vse $k \geq 1$ velja

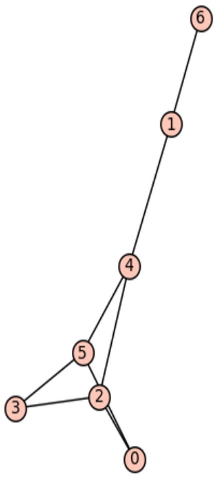
$$\gamma_{krt}(G) = \gamma_t(G \Box K_k).$$

3 Reševanje problema

3.1 Majhni grafi

```
result=[]; #seznam vseh gamak/gama
gresult=[]; #seznam proti primerov
n=8; #število vozlišč
p=0.5; #verjetnost da posamezno povezavo dodaš v graf
for i in range(100): #preveri na 10ih random grafih
    for k in range(4,6):
        g=graphs.RandomGNP(n,p)
        gk=g.cartesian_product(graphs.CompleteGraph(k))
        gama=g.dominating_set(value_only=True)
        gamak=gk.dominating_set(value_only=True,total=True)
        x=float(gamak/gama);
        if x<2: gresult.append(g)
        if x==2: dvaresult.append(g)
        result.append(x)
print(min(result))
```

```
dvaresult[0].plot() #primer grafa, ko je koeficient enak 2
```




```
#trivialen primer, kjer je koeficient enak 2
for G in graphs(7, size=0):
    G.show()

(G.cartesian_product(graphs.CompleteGraph(7))).dominating_set(value_only=True, total=True)/G.dominating_se
t(value_only=True)
```

```
k = 4
def spremeni_graf(G):
    H = Graph(G)
    if random() < 0.5:
        H.delete_edge(H.random_edge())
        if is_connected(H):
            H
        else:
            H = Graph(G)
    else:
        if H.complement().size() == 0:
            H.delete_edge(H.random_edge())
        else:
            H.add_edge(H.complement().random_edge())
    return H
```

```
def krit(G):
    return (G.cartesian_product(graphs.CompleteGraph(k))).dominating_set(value_only=True, total=True)/G.do
minating_set(value_only=True)
```

```
while True:
    G_1 = graphs.RandomGNP(7, 0.5)
    if G_1.is_connected():
        G = G_1
        break
    else:
        True
G.show()
```