UDRUGA HRVATSKIH GRAĐEVINSKIH FAKULTETA
SVEUČILIŠTE U ZAGREBU GRAĐEVINSKI FAKULTET
ZAJEDNIČKI TEMELJI '17.

DOI: https://doi.org/10.5592/CO/ZT.2017.22

# Konstrukcija maksimalno balansirane povezane particije u grafu

**Anton Vrdoljak**

Sveučilište u Mostaru, Građevinski fakultet
*kontakt: anton.vrdoljak@gfmo.ba*

## Sažetak

Implementacije mnogih metoda diskretne matematike, posebice teorije grafova i teorije kompleksnih mreža, te metode rudarstva podataka, imaju važnu ulogu u rješavanju NP – teških problema iz područja kombinatorne optimizacije. Dosadašnja istraživanja su pokazala kako značajnu ulogu o ovim problemima imaju istraživanja važnosti pojedinog vrha unutar grafa, te konstruiranje maksimalno balansirane particije u grafu. Takvi problemi u praksi nisu rijetki, ali nisu ni predmet istraživanja isključivo matematičara, već ih izučavaju i istraživači iz drugih područja znanosti, inženjerstva, ekonomije, obrazovanja.

***Ključne riječi:*** *graf, maksimalno balansirana povezana particija u grafu, pohlepni algoritam, cluster analiza*

# Structure of maximally balanced connected partition in graph

## Abstract

The implementation of various discrete mathematics methods, especially graphs, complex network theory, and the data mining methods, play an important role in solving NP–difficult problems in the field of combinatorial optimization. Previous research has shown that these issues are greatly influenced by study of the importance of a single vertex within the graph, and the design of maximally balanced connected partitions in graph. These problems are not rare in practice, and are not studied solely by mathematicians, but also by researches from other areas of science, engineering, economy, and education.

***Keywords:*** *graph, maximally balanced connected partition in graph, greedy algorithm, cluster analysis*

# 1 Uvod

## 1.1 Uvodne definicije

Mnogi se problemi u suvremenoj znanosti mogu modelirati uz pomoć grafova, i vrlo često se svode na nalaženje putova unutar grafova. Ugrubo, graf se može predstaviti kao familija točaka, koje nazivamo vrhovima, zajedno sa spojnicama između točaka, koje nazivamo bridovima. Precizna definicija grafa glasi [1]:

**Graf** je uređena trojka G = (V, E, φ), gdje je V neprazan skup vrhova, E skup bridova koji je disjunktan s V, a φ funkcija koja svakom bridu iz E pridružuje dva, ne nužno različita vrha iz V. Graf često zapisujemo kao par G = (V, E) ili samo G.
Svaki brid e ∈ E spaja dva vrha u, v ∈ V. Kažemo još kako su u i v krajevi brida e, te da su u i v susjedni vrhovi, i pišemo $e = uv$. Bridovi čiji se krajevi podudaraju zovu se petlje, a bridovi kod kojih su krajevi različiti zovu se pravi bridovi ili karike. U ovom radu ćemo proučavati samo konačne grafove. Konačan graf je onaj u kojem su V i E konačni skupovi. Graf je jednostavan ako nema petlje i ako ne postoje dva brida koji spajaju isti par vrhova [2]. Graf koji može sadržavati petlje i višestruke bridove nazivamo pseudograf.
**Podgraf** grafa $G = (V, E)$ je svaki graf $G' = (V', E')$ sa svojstvom: $V' \subseteq V$ i $E' \subseteq E$. To još zapisujemo s $G' \subseteq$ Podgraf $G'$ grafa $G$ sa svojstvom $V(G') = V(G)$ naziva se **razapinjući podgraf** grafa $G$. Također, neka je $V' \subseteq V$ neprazan podskup vrhova grafa $G$, podgraf grafa $G$ čiji je skup vrhova $V'$, a skup bridova je podskup skupa bridova grafa $G$ i sastoji se od onih bridova iz $E$ čija su oba kraja u $V'$ naziva se **podgraf induciran** s $V'$. Nadalje, put se definira kao poseban slučaj šetnje [1, 2].
Šetnja $W$ u grafu $G = (V, E)$ je netrivijalni konačni niz $v_0e_1v_1e_2v_2...v_{m-1}e_mv_m$, čiji su članovi naizmjenice vrhovi i bridovi grafa Vrhovi su na početku i na kraju šetnje. Vrhovi koji nisu na početku ili kraju, zovu se **unutarnji vrhovi**. **Duljina šetnje** $k$ je broj bridova u šetnji. Ako su svi vrhovi u šetnji međusobno različiti, šetnju nazivamo **put**.
Pseudograf je **povezan** ako i samo ako postoji šetnja između bilo koja dva njegova vrha.
Inače, pišemo $v_1 \sim v_2$ ako postoji šetnja u grafu koja povezuje vrhove $v_1$ i $v_2$ tog grafa. Budući da je ~ relacija ekvivalencije na skupu $V$ svih vrhova grafa $G$, na skupu $V$ postoji particija tog skupa na disjunktne, neprazne pod skupove – klase (ekvivalencije).
Graf $G$ u kojem se skup vrhova može particionirati u $k$ nezavisnih skupova zovemo ***k*–partitivnim** ili **višepartitivnim** grafom s $k$ nezavisnih skupova.
$k$–partitivni graf $G$ je **potpun** ako za bilo koja dva vrha grafa $G$ vrijedi da su susjedni ako i samo ako pripadaju različitim klasama particije skupa vrhova $V$ grafa $G$.

### 1.2 Matematička definicija problema

Sada ćemo dati matematičku definiciju problema:
Neka je $G = (V, E)$, $V = \{1, 2, \ldots, n\}$ povezan graf i $|E| = m$. Neka su $w_i$ težine u vrhovima. Za proizvoljan $V' \subseteq V$ uvedimo oznaku $w(V')$ za sumu težina svih vrhova (čvorova) iz $V'$, tj.

$$w\left(V'\right) = \sum_{i \in V'} w_i \tag{1}$$

**Problem maksimalno balansirane povezane particije u grafu** jest određivanje particije skupa vrhova iz $V' \subseteq V$ u dva disjunktna, neprazna skupa, $V_1$ i $V_2$, takva da su podgrafovi grafa $G$ inducirani s $V_1$ i $V_2$ povezani, a sume težina vrhova iz $V_1$ i $V_2$ su po vrijednosti što bliže jedna drugoj, tj. razlika između suma težina je najmanja moguća.

## 2 Algoritmi za konstrukciju maksimalno balansirane povezane particije

U ovom radu primijenjene su dvije strategije (algoritma, tehnike) za konstrukciju maksimalno balansirane povezane particije u grafu: **pohlepni algoritam**, tzv. *Greedy*, i **cluster analiza**.

### 2.1 Pohlepni algoritam

Izbor pohlepnog algoritma se nametnuo zbog činjenice što takvi algoritmi koriste metaheuristiku za rješavanje problema, te su obično jednostavnog oblika i daju brzu aproksimaciju najboljega rješenja. Prema Singeru [3], opći oblik pohlepnog algoritma je:

```
procedure greedy (C : skup; var S : skup; var OK : boolean)
        { C je u početku skup svih raspoloživih kandidata. }
        { U skupu S akumuliramo rješenje problema. }
        { OK kaže da li je nađeni S rješenje. }
begin
        S := 0;
while not rješenje(S) and (C ≠ Ø) do
        begin
        x := element iz C koji maksimizira vrijednost izbor(x);
        C := C\{x};
        if dopustivo(S ∪ {x}) then S := S ∪ {x};
        end;
OK := rješenje(S);
end.
```

Nadalje, isti autor [3] ističe kako pohlepni algoritam **ne** mora naći rješenje problema. Čak i ako ga nađe, to rješenje **ne** mora biti optimalno, jer pohlepni algoritam nigdje ne koristi funkciju cilja, već samo funkciju izbora. Da bi ovakvo nađeno rješenje bilo i optimalno rješenje, treba **dokazati** da pohlepni algoritam korektno rješava problem optimizacije.

## 2.2 Cluster analiza

Prema Vukičeviću [4] Cluster analiza nije statistička tehnika, već skup metoda (algoritama) koji nam omogućuju da klasificiramo promatrane podatke. Što je promatrana klasa manja, to su razlike među objektima manje, a što je promatrana klasa veća, to su razlike među objektima veće. Dakle, možemo reći kako Cluster analiza za svrhu ima otkrivanje strukture nekog početnog skupa podataka (npr. gradiva ili sadržaja nekog predmeta na studiju) njegovim dekomponiranjem (particioniranjem) u manje, homogenije podskupove.
Prvi problem Cluster analize je kako utvrditi (tj. brojčano izraziti) sličnost između dvaju objekata *a* i *b* opisanih nekim svojstvima $s_1, s_2, \ldots, s_n$. Neka je vrijednost svojstva $s_i$ za objekt *a* jednaka $x_i$, a za objekt *b* jednaka $y_i$. Tada se neka od *udaljenosti* (s obzirom na sličnost) objekata *a* i *b* može mjeriti na više načina.

1) Euklidska udaljenost:

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2}$$

2) Kvadrirana euklidska udaljenost:

$$d(a,b) = \sum_{i=1}^{n}(x_i - y_i)^2 \tag{3}$$

3) Manhattan ili taxi udaljenost:

$$d(a,b) = \sum_{i=1}^{n} | x_i - y_i | \tag{4}$$

4) Čebiševljeva udaljenost:

$$d(a,b) = max | x_i - y_i | \tag{5}$$

5) Minkowski udaljenost:

$$d(a,b) = \sqrt[p]{\sum_{i=1}^{n} | x_i - y_i |^p} \tag{6}$$

6) Udaljenost s obzirom na postotak neslaganja:

$$d(a,b) = card\{i : x_i \neq y_i\} \tag{7}$$

U ovom radu je korištena samo euklidska udaljenost. Nadalje, potrebno je utvrditi kako formirati klastere, te kako utvrditi konačan broj klastera. Prema matematičkoj definiciji problema, konačan broj klastera je dva. Inače, uvriježeno je kako istraživač sam treba prosuditi, u kontekstu svog istraživanja, koji broj klastera i s kakvim karakteristikama mu je potreban. Dakle, preostaje riješiti pitanje utvrđivanja udaljenosti dvaju skupova objekata, a postoji nekoliko načina na koje to možemo učiniti, primjerice,
1) jednostrukom povezanošću:

$$d(A,B) = \min\{d(a,b) : a \in A, b \in B\} \tag{8}$$

2) potpunom povezanošću:

$$d(A,B) = \max\{d(a,b) : a \in A, b \in B\} \tag{9}$$

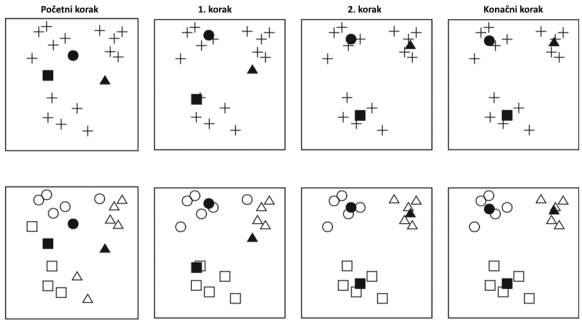
3) neuteženom prosječnom povezanošću:

$$d(A,B) = \left( \sum_{a \in A, b \in B} d(a,b) \right) / (cardA \cdot cardB) \tag{10}$$

Nakon što je odabran način na koji ćemo mjeriti udaljenost među skupovima objekata, te nakon što je utvrđen konačan broj klastera, prema [4] primijenit ćemo sljedeći algoritam:

1) Ako se u familiji skupova nalaze barem dva skupa,
   1.1) pronađi dva skupa kojima je udaljenost najmanja
   1.2) smjesti ta dva skupa u isti klaster
   1.3) izbaci ta dva skupa iz familije promatranih skupova
   1.4) u familiju promatranih skupova dodaj skup koji je unija dva izbačena skupa
   1.5) vrati se na liniju 1).
2) Ako se u familiji skupova nalazi samo jedan skup,
   2.1) kraj algoritma.

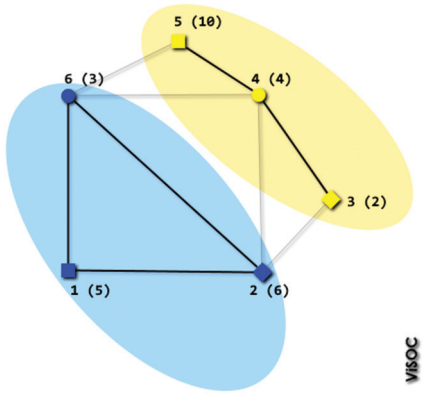
Važno je napomenuti kako različiti izbori računanja udaljenosti među objektima i među skupovima rezultiraju različitim klasteriranjem objekata. Jedan primjer tehnike algoritma cluster analize prikazan je na slici 1.




**Slika 1. Tehnika algoritma cluster analize**

## 3 Implementacija

Postupak rješavanja nekog problema uz uporabu računala obično se sastoji od sljedeće tri faze: **formulacija problema**, odnosno formiranje matematičkog problema, **definicija** postupka, odnosno **algoritma rješavanja**, koji razumijeva precizan niz radnji koje treba obaviti kako bi se došlo do rješenja problema, te **implementacija**, odnosno **programiranje**, danog algoritma u nekom od (viših) programskih jezika. Prilikom definiranja algoritma preporučuje se uzeti u obzir da će taj problem biti riješen uz pomoć računala, i da će taj algoritam sadržavati jednu ili više **numeričkih metoda**, pa se trebaju osigurati određeni ulazni podaci, izvršiti odgovarajuća testiranja i predvidjeti u kojoj će formi biti prezentirani rezultati proračuna. Jasno je kako postoji niz međudjelovanja ovih triju faza, jer kako se program razvija, dobiva se sve više informacija o problemu koje mogu sugerirati određene promjene u formulaciji problema, u usvojenom algoritmu, ili u programu. Implementacija algoritama za konstrukciju maksimalno balansirane povezane particije u grafu izvedena je u objektno orijentiranom programskom jeziku C#, unutar softverskog paketa Microsoft Visual Studio 2015. Izrađen je programski modul GCmaxPart, prema shemi [8], koji u pripremnoj fazi konstruira graf, tj. zadaje vrhove i bridove (veze) te težine u vrhovima. Nakon toga slijedi izvršna faza u kojoj će korisnik imati mogućnost konstrukcije maksimalno balansirane povezane particije u grafu pomoću dviju (u točki dva) opisanih strategija (pohlepni algoritam i cluster analiza) i imati, na samom kraju, mogućnost validacije dobivenih rezultata (ispitivanje je li dobivena particija odgovarajuća) kao i evaluacije čitavog postupka. Dobiveni rezultati nedvojbeno pokazuju kako su implementirane modifikacije strategija (tehnika) pogodne za konstrukciju maksimalno balansirane povezane particije u grafu. Na slici 2. prikazan je jednostavan primjer (malog) grafa i maksimalno balansirane povezane particije u njemu.




**Slika 2. Graf i rješenje problema maksimalno balansirane povezane particije u njemu [8]**

## 4 Zaključak

Problem konstrukcije maksimalno balansirane povezane particije u grafu vezan je uz teoriju grafova, a ima čestu primjenu u znanosti, posebice u inženjerstvu, ekonomiji i, u posljednje vrijeme sve češće, obrazovanju (organiziranje lekcija tečaja u dvije povezane cjeline koje su ujednačene po težini [8]). Budući da kardinalnost skupa vrhova $V$ u grafu $G$ može biti velika i budući da je poznato kako se problem maksimalno balansirane povezane particije u grafu karakterizira kao NP–težak problem (vrlo zahtjevan za rješavanje), logično je bilo odlučiti se pribjeći korištenju pohlepnih algoritama i tehnika cluster analize u konstrukciji. Provedena implementacija, u obliku programskog modula GCmaxPart, daje izuzetno efikasne konstrukcije u razumnom vremenu (složenost algoritma je $O(n^2)$).

## Literatura

[1] Kalebić, F., Mandić, J., Vukičević, D., Braić, S.: Prebrojavanje savršenih sparivanja, Hrvatski matematički elektronički časopis math.e, 27 (2009).

[2] Divjak, B., Lovrenčić, A.: Diskretna matematika s teorijom grafova, Sveučilište u Zagrebu, FOI, Varaždin, 2005.

[3] Singer, S.: Složenost algoritama, Sveučilište u Zagrebu, PMF – Matematički odjel, Zagreb, 2005.

[4] Vukičević, D.: Uvod u statistiku, Sveučilište u Splitu, Split, 2005.

[5] MacKay, D.: Information Theory, Inference, and Learning Algorithms, Version 7.2, Cambridge University Press, Cambridge, 2005.

[6] McMillan, M.: Data Structures and Algorithms Using C#, Cambridge University Press, Cambridge, 2007.

[7] Newman, M.: Networks, An Introduction, Oxford University Press, Oxford, 2010.

[8] Matić, D., Božić, M.: Rješavanje nekih organizacionih problema u nastavi primjenom pronalaženja maksimalno balansirane povezane particije u grafu, SIMPOZIJUM Matematika i primene, Mateljević M., Beograd, Univerzitet u Beogradu, Matematički fakultet, (N61 – N67), 2012.