PROGRAMSKO PRONALAŽENJE RJEŠENJA MBCP PROBLEMA

mr. sc. **Anton Vrdoljak**, prof. matematike
Građevinski fakultet Sveučilišta u Mostaru

Sažetak: U radu je dan osvrt na programsko rješavanje problema vezanog za particioniranje grafa *G = (V, E)*. Budući je to problem koji pripada klasi NP–teških problema odabrane su strategije iz domene heuristike za njegovo rješavanje. U tu svrhu napravljen je programski modul u softverskom paketu Microsoft Visual Studio 2015. Implementirani modul programski (heuristika) pronalazi sub–optimalna rješenja, a nerijetko i korektno optimalno rješenje MBCP problema korisničkog grafa u razumnom vremenu.

Ključne riječi: povezan graf, particioniranje grafa, MBCP, pohlepni algoritam, cluster analiza

PROGRAMMATICALLY SOLUTION FOR MBCP PROBLEMS

Abstract: The paper gives a review regarding the problem solving of the graph partitioning problem, for some graph *G = (V, E)*. Since this is a problem belonging to the NP-Hard class, strategies from heuristic domain were chosen for solving such a problem. For this purpose, a software module was created in the Microsoft Visual Studio 2015 software package. The implemented module programmatically (heuristics) will find the sub–optimal solutions, but also finding of the correct optimum for MBCP problem of user's graph in reasonable time is not rare.

Key words: connected graph, graph partitioning, MBCP, greedy algorithm, cluster analysis

1. UVOD

U fokusu ovog rada je programsko rješenje za problem pronalaženja maksimalno balansirane povezane particije u grafu (engl. Maximally Balanced Connected Partition Problem – MBCP). Izrađen je programski modul u objektno orijentiranom programskom jeziku C#, unutar softverskog paketa Microsoft Visual Studio 2015. Uvodne definicije za graf, inducirani podgraf, šetnju, put, povezani graf i k–partitivni graf su izostavljene, i mogu se pronaći u literaturi [4].

MBCP problem pripada široj klasi problema koji se tiču particioniranja grafa. Kako ovi problemi spadaju u kombinatornu optimizaciju, područje koja sadrži mnoge NP–teške probleme (problemi za koje ne postoji algoritam koji pronalazi rješenje u polinomijalnom vremenu), nametnula je se potreba za korištenjem neke od heurističkih metoda. Zato je u ovom radu korištena strategija pohlepnog algoritma, tzv. *Greedy*, za programsko pronalaženje rješenja MCBP problema. Pored ovog pristupa primjenjuje se i strategija cluster analize. Razlog primjene cluster analize je u svrsi ovog skupa metoda (algoritama), a on je otkrivanje strukture nekog početnog skupa podataka njegovim dekomponiranjem (particioniranjem ili klasificiranjem) u manje, homogenije podskupove (clustere), koji su uzajamno isključivi (disjunktni).

Po definiciji heuristika (prema grčki εὑρίσκειν: nalaziti, otkrivati) je postupak koji vodi prema otkriću ili ga potiče. U modernoj logici heuristika se opisuje kao proces koji može riješiti određenu vrstu problema, ali ne jamči uspješno rješenje. To je značenje preuzeto u suvremenu filozofiju uma, u kojoj se termin heuristika primjenjuje na formalno neispravne i/ili nepotpune procedure zaključivanja ili odlučivanja. Takve procedure često vode do uspješnoga rješenja problema, ako se primijene na specifično i razmjerno usko područje, no, primijenjene na neko drugo područje pokazuju se neuspješnima i beskorisnima. Drugim riječima, kognitivna heuristička „pravila“ pri odlučivanju su lakši put do donošenja odluka, često vrlo efikasna, ali ne vode uvijek najboljoj opciji.

Formulacija problema, odnosno formiranje matematičkog problema u pravilu predstavlja prvu fazu, od ukupno tri faze, u postupcima za pronalaženje rješenja nekog problema uz uporabu računala. U drugoj fazi definira se postupak, odnosno algoritam rješavanja, koji razumijeva precizan niz radnji koje treba obaviti kako bi se došlo do rješenja problema. Posljednja faza je implementacija, odnosno programiranje, danog algoritma u nekom od (viših) programskih jezika. Nužno je biti svjestan činjenice o uskoj povezanosti ovih triju faza, odnosno o postojanju niza 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 [4]. U paragrafu (podnaslovu) koji slijedi daje se matematička definicija problema.

1.1 Matematička definicija problema

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

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

MBCP problem jest problem pronalaženja (određivanja) particije skupa vrhova V grafa G u dva disjunktna 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.

1.2 Zašto je particioniranje teško?

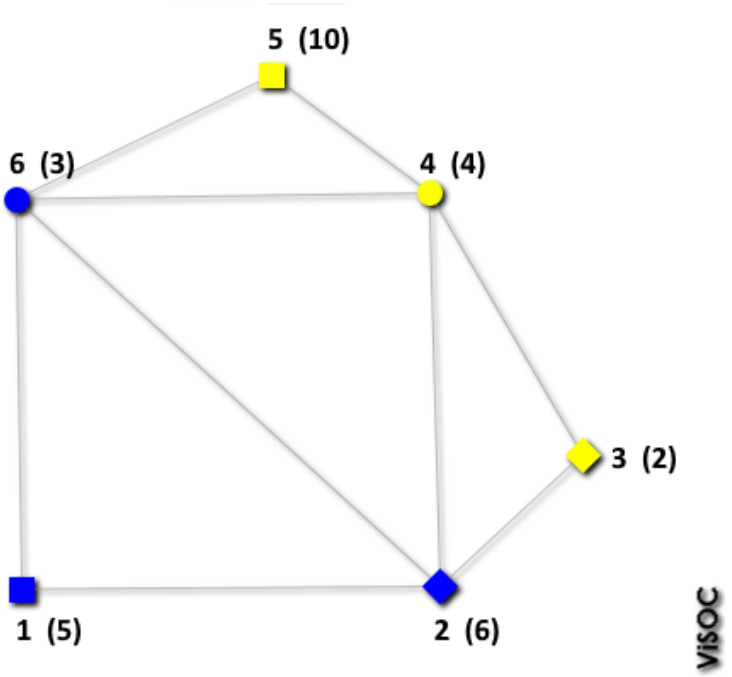
Najjednostavniji problem particioniranja grafa bi bio podjela tog grafa u samo dva dijela (bisekcija grafa). Velika većina poznatih algoritama za particioniranje grafa upravo su algoritmi za bisekciju grafa, a tek nekoliko njih za rezultat daje podjelu grafa u proizvoljan broj dijelova (prebrojivo mnogo dijelova). Ovo se na prvi pogled može činiti nedostatkom, ali u praksi to nije, jer ako možemo podijeliti graf na dva dijela, onda ga možemo podijeliti i na više od dva dijela, daljnjim dijeljenjem jednog ili oba inicijalna dijela. Ovakav pristup ponavljanja bisekcije je najčešće korišteni pristup za podjelu grafa u proizvoljan broj dijelova. Iako ga je relativno jednostavno opisati, ovaj problem nije lagan za riješiti. Jedan od mogućih pristupa u postupku rješavanja bi bio podjela grafa u dva dijela jednostavnim pregledavanjem svih mogućih podjela grafa u dva dijela, te zatim odabiranjem one podjele koja zadovoljava dane uvjete. Međutim, za sve grafove osim za tzv. male grafove, takva tzv. iscrpna pretraga biva neizmjerno skupa u smislu vremena računanja [8].

Broj načina na koji možemo izvršiti podjelu grafa sa n vrhova na dva dijela (podgrafa) sa n_1 i n_2 vrhova je $n!/(n_1!\ n_2!)$. Ako aproksimiramo faktorijele pomoću Stirlingove formule, i iskoristimo činjenicu da je $n_1 + n_2 = n$, dobivamo

$$\frac{n!}{n_1!\,n_2!} \approx \frac{\sqrt{2\pi n}\,(n/e)^n}{\sqrt{2\pi n_1}\,(n_1/e)^{n_1}\sqrt{2\pi n_2}\,(n_2/e)^{n_2}} \approx \frac{n^{n+1/2}}{n_1^{n_1+1/2}\,n_2^{n_2+1/2}}. \tag{2}$$

Tako, primjerice, ako želimo podijeliti graf na dva dijela jednake veličine ($n_1 = n_2 = n/2$), broj različitih načina za to je približno

$$\frac{n^{n+1/2}}{(n/2)^{n+1}} = \frac{2^{n+1}}{\sqrt{n}}. \tag{3}$$



Slika 1. Mali graf sa šest vrhova i devet bridova

Dakle, količina vremena potrebnog za promatranje svih mogućih podjela grafa u dva dijela će rasti približno eksponencijalno s povećanjem kardinalnosti skupa vrhova *V* u grafu *G*. Nažalost, riječ je o vrlo brzoj rastućoj funkciji, što povlači da opisani pristup podjele grafa u samo dva dijela nema velikog smisla u realnom vremenu, čak i za prilično umjerene vrijednosti od *n*. Potonji zaključci, osim što su dali odgovor na postavljeno pitanje u ovom podnaslovu, daju nam i potvrdu logike opravdanosti odluke da se pribjegne korištenju pohlepnih algoritama i tehnika cluster analize pri implementaciji programskog pronalaženja rješenja MCBP problema.

2. ALGORITMI ZA PRONALAŽENJE RJEŠENJA MBCP PROBLEMA

U uvodu je spomenuto kako su u ovom radu primijenjene dvije strategije (algoritma, tehnike) za programsko pronalaženje rješenja MCBP problema: pohlepni algoritam i cluster analiza.

Pohlepni algoritmi, tzv. *Greedy*, koriste metaheuristiku za rješavanje problema, te su u pravilu jednostavnog oblika (koda) i daju brzu aproksimaciju najboljeg rješenja. Prema Singeru [5] pohlepni algoritam ne mora naći rješenje problema. Nadalje, 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. Opći oblik pohlepnog algoritma je prikazan u tablici ispod.

Tablica 1. Opći oblik pohlepnog algoritma prema Singeru [5]

```
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 rjesenje(S) and (C ≠ ∅) do
                        begin
                        x := element iz C koji maksimizira vrijednost izbor(x);
                        C := C \ {x};
                        if dopustiv(S ∪ {x}) then S := S ∪ {x};
                        end;
                OK := rjesenje(S);
end; { greedy }
```

Cluster analiza je 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. Drugim riječima, cluster treba imati visoku unutarnju (unutar clustera) homogenost i visoku vanjsku (između clustera) heterogenost.

Prvi problem cluster analize je kako utvrditi (tj. brojčano izraziti) sličnost između dvaju objekata *a* i *b* opisane nekim svojstvima s_1, s_2, …, s_n. Neka je vrijednost svojstva s_i za objekt *a* jednaka x_i, a za objekt *b* jednaka y_i. Tada udaljenost (obzirom na sličnost) objekata *a* i *b* možemo mjeriti na više načina, a u ovom radu je korištena samo euklidska udaljenost:

$$d(a,b) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$

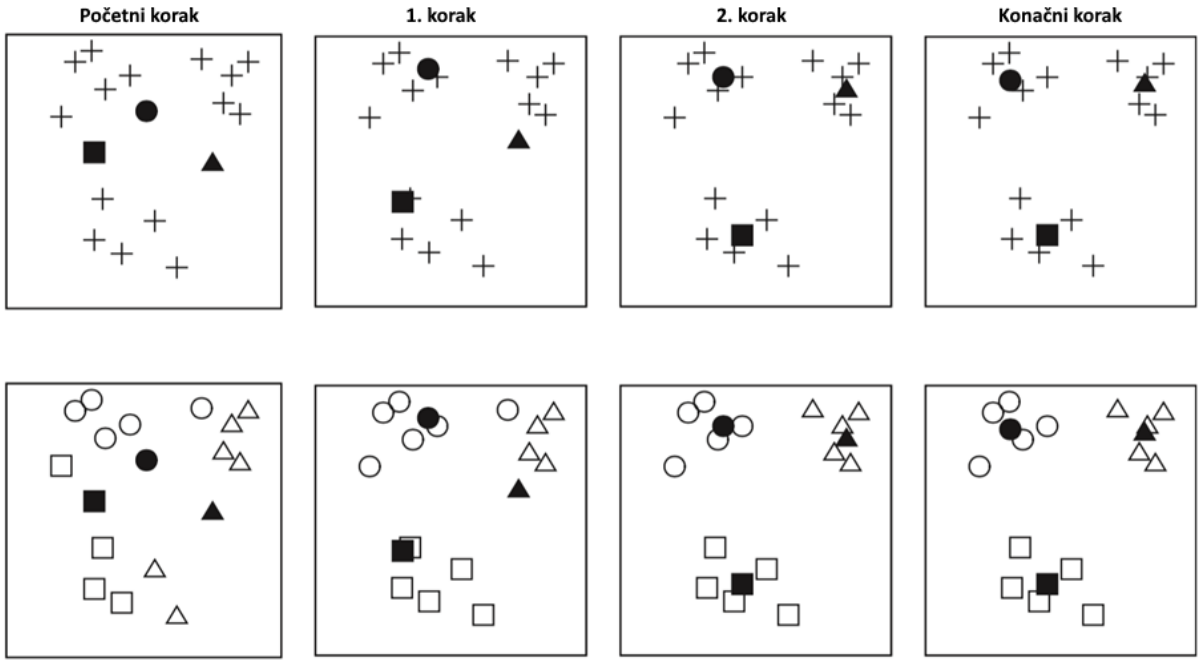
Nadalje, potrebno je utvrditi kako formirati clustere, te kako utvrditi konačan broj clustera. Prema matematičkoj definiciji problema u ovom radu, konačan broj clustera je dva. Dakle, preostaje riješiti pitanje utvrđivanja usaljenosti dvaju skupova objekata. Postoji nekoliko načina na koje to možemo učiniti, a navest ćemo samo neke od njih [4]: jednostruka povezanost (udaljenost među skupovima *A* i *B* je najmanja udaljenost među objektima *a* i *b* takvima da je *a* iz *A*, a *b* iz *B*), potpuna povezanost (udaljenost među skupovima *A* i *B* je najveća udaljenost među objektima *a* i *b* takvima da je *a* iz *A*, a *b* iz *B*)), neutežena prosječna povezanost (udaljenost među skupovima *A* i *B* je prosječna udaljenost među objektima *a* i *b* takvima da je *a* iz *A*, a *b* iz *B*).

Nakon što je odabran način na koji ćemo mjeriti udaljenost među skupovima objekata (clusterima), te nakon što je odabran konačan broj clustera, primijenit ćemo algoritam prikazan u tablici ispod.

Tablica 2. Algoritam cluster analize [4]

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 cluster*
	1.3) *izbaci ta dva skupa iz familije promatranih skupova*
	1.4) *u familiji 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 objekata rezultiraju različitim clusteriranjem objekata. Jedan primjer tehnike algoritma cluster analize prikazan je na slici 2.



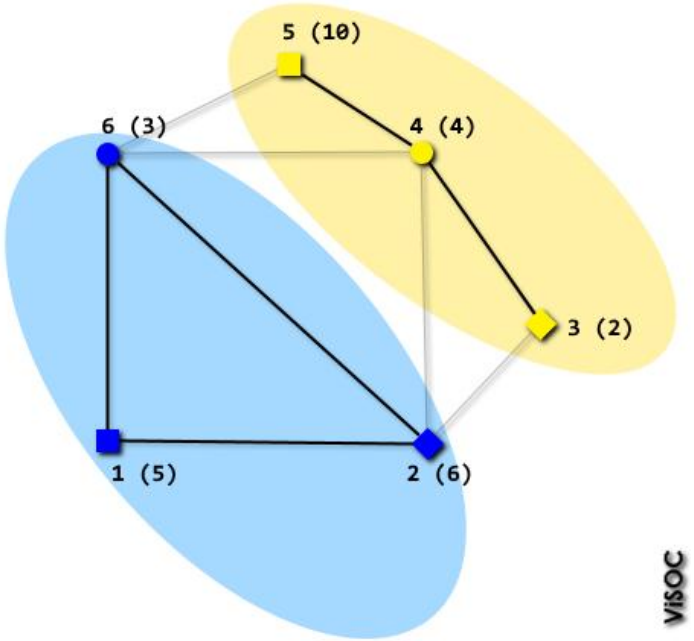
Slika 2. Tehnika algoritma cluster analize

3. PROGRAMSKI MODUL

Implementacija programskog pronalaženja rješenja MCBP problema izvedena je u objektno orijentiranom programskom jeziku C#, unutar softverskog paketa MS Visual Studio 2015. Izrađen je programski modul koji u pripremnoj fazi omogućuje korisniku konstrukciju grafa, tj. zadavanje vrhova i bridova (veze među vrhovima) te težine u svim vrhovima grafa. U svakom trenutku je moguće spremanje grafa u tekstualnu datoteku, kao i ispis grafa u .pdf datoteku. Po završetku konstruiranja grafa slijedi izvršna faza u kojoj će korisnik imati mogućnost pronalaženja rješenja MBCP problema za upravo konstruiran graf. Za izbor će imati dvije (u prethodnom naslovu) opisane strategije (pohlepni algoritam i cluster analiza). Pseudo–kod implementiranog pohlepnog algoritma dobro je poznat u literaturi [1, 3] a naš je prikazan u tablici ispod. Programskim modulom još su predviđene dvije instrukcije: mogućnost validacije dobivenih rezultata kao i evaluacija čitavog postupka. Na slici 3. prikazan je jednostavan primjer (malog) grafa s jednim optimalnim rješenjem MBCP problema tog grafa, koje neće biti pronađeno uz pomoć ovog programskog modula.

Tablica 3. Pohlepni algoritam za pronalaženje rješenja MBCP problema

1)	*Sortiramo vrhove iz skupa V tako da vrijedi* $w(v_1) \geq w(v_2) \geq \ldots \geq w(v_n)$,
2)	*počinjemo sa* $V_1 := \{v_1\}$, $V_2 := V \setminus V_1$,
3)	*ako je* $w(V_1) \geq (1/2)w(V)$
	3.1) *idi na liniju* **7)**, *u suprotnom idi na liniju* **4**),
4)	*formiraj skup* $V_0 = \{u \in V_2 \mid (V_1 \cup \{u\}, V_2 \setminus \{u\})$ *je povezana particija grafa G* },
5)	*izaberi vrh* $u \in V_0$ *takav da je* $w(u) = min \{w(v) \mid v \in V_0\}$,
6)	*ako je* $w(u) < w(V) - 2w(V_1)$
	6.1) *tada je* $V_1 := V_1 \cup \{u\}$, $V_2 := V_2 \setminus \{u\}$,
	6.2) *vrati se na liniju* **3)**, *u suprotnom idi na liniju* **7)**,
7)	*prikaži particiju* (V_1, V_2).



Slika 3. Mali graf sa jednim optimalnim rješenjem MCBP problema

3. ZAKLJUČAK

MBCP problemi, odnosno pronalaženja njihovih rješenja, spadaju u kombinatornu optimizaciju, te su u uskoj vezi s teorijom grafova. Imaju vrlo čestu primjenu u znanosti, posebice inženjerstvu i spadaju u klasu NP–problema (jer kardinalnost skupa vrhova *V* u promatranom grafu *G* može biti velika). Izrađeni programski modul koji pronalazi sub–optimalno rješenje MBCP problema koristi heuristiku, ali i ponekad daje korektne particije u razumnom vremenu (složenost algoritma je $O(n^2)$). Mogućnost daljnjeg razvoja programskog modula ostaje u području uvođenja dodatnih metoda heuristike koje su poznate u literaturi (npr. genetički algoritam, pretraga promjenjivom okolinom), ili uvođenja potpuno novih heurističkih metoda, s ciljem poboljšanja kvalitete rješenja.

LITERATURA

1. Chlebikova, J.: *Approximating the Maximally Balanced Connected Partition Problem in Graphs*, Information Processing Letters, 1996, 60, str. 225-230
2. Djurić, B. i ostali: *Solving the Maximally Balanced Connected Partition in Graphs by Using Genetic Algorithm*, Computers and Informatics, 2008, 27(3), str. 341-354
3. Matić, D., Božić, M.: *Maximally Balanced Connected Partition in Graphs: Application in Education*, Teaching of Mathematics, 2012, 15(2), str. 121-132
4. Vrdoljak, A.: *Konstrukcija maksimalno balansirane povezane particije u grafu*, Zbornik radova 5. skupa mladih istraživača – ZAJEDNIČKI TEMELJI 2017., 2017, 5, str. 129-136
5. Singer, S.: *Složenost algoritama*, Sveučilište u Zagrebu, PMF – Matematički odjel, Zagreb 2005.
6. MacKay, D.: *Information Theory, Inference, and Learning Algorithms*, *Version 7.2*, Cambridge University Press, Cambridge 2005.
7. McMillan, M.: *Data Structures and Algorithms Using C#*, Cambridge University Press, Cambridge 2007.
8. Newman, M.: *Networks, An Introduction*, Oxford University Press, Oxford, 2010.