
# Računarska inteligencija

Rešavanje problema minimalnog pokrivača čvorova

Tijana Tošev 256/2015
Jovana Nikolić 3/2015
Matematički fakultet

# Sadržaj

# Uvod

Tema ovog seminarskog rada je problem minimalnog pokrivača grafa. U delu formulacije problema upoznaćemo se sa detaljima ovog problema. Zatim sledi dokumntacija implementacije u kojoj je opisano nekoliko algoritama za rešavanje.

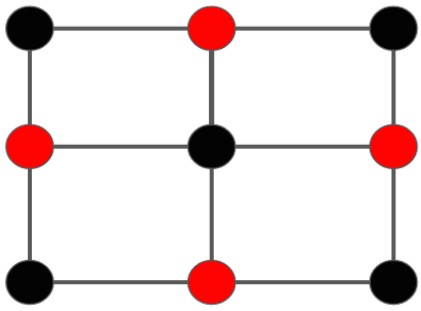
U poglavlju analiza vrši se poređenje dobijenih rezultata iz poznatih i prethodno implemnetirani rešenja. Zaključak govori o primenama algoritama i daljim planovima za unapređivanje. Na kraju ovog rada navedena je sva korišćena literatura za njegovu izradu.

### Formulacija problema

Pokrivač grana neusmerenog grafa G = (V, E) je skup čvorova U takav da je svaka grana iz E susedna bar jednom čvoru iz U.

Ulaz: graf G = (V, E), gde je V skup čvorova a E skup grana koje čine graf G

Izlaz: minimalan skup cvorova V’⊆V, takav da za svaku ivicu (A,B)∈E bar jedan od cvorova A ili B pripada V’.



*slika 1 - crveni cvorovi predstavljaju minimalan pokrivac grana, a crni pokrivac grana koji nije minimalan*

*Teorema: Problem pokrivača grana je NP-kompletan.*

*Dokaz:* Za neko k∈N za koje smo pretpostavili da je rešenje koje predstavlja broj čvorova kojih pripadaju pokrivaču grafa možemo proveriti u polinomijalnom vremenu da li je rešenje problema minimalnog pokrivača. Da bi dokazali da je problem NP-kompletan treba da na njega svedemo neki poznati NP-kompletan problem. U ovom dokazu svodićemo problem klike. U nekom neusmerenom grafu G klika je podgraf grafa G u kome su svi čvorovi međusobno povezani granama iz G. Potrebno je utvrditi da li u datom grafu postoji klika veličine k. Posmatramo proizvoljni ulaz za problem klike G = (V, E),k i komplement datog ulaznog grafa G’ = (V, E’).

Kardinalnost čvorova predstavimo sa n = |V|. Dalje, treba dokazati da ulaz za problem klike (G,k) sveden na ulaz za nas problem u obliku (G', n − k). Posmatrajmo kliku C = (U, F) u G. Skup čvorova V \ U pokriva sve grane u G', jer u G' nema grana koje povezuju čvorove iz U (sve te grane su u G). Iz ovog zaključka sledi da je V \ U pokrivač grana za G'. Ako G ima kliku veličine k, onda G' ima pokrivač grana veličine n−k. Dokažimo sada obrnutu stranu. Neka je D pokrivač grana u G'. Tada D pokriva sve grane u G', pa u G' ne postoji grana koja povezuje neka dva čvora iz V \ D. Znači V \ D je klika u G. Ovim je dokazano da ako u G' postoji pokrivač grana velicine n−k, onda u G postoji klika veličine k. Ova redukcija se vrši u polonimijalnom vremenu. ¤

Može se napomenuti da u nekim specijalnim slučajevima algoritam ima polinomijalnu složenost. Naime za slučaj bipartitnog grafa ovaj problem se svodi na polinomijalan problem uparivanja grana. Takođe ako graf predstavlja stablo tada je pomocu pohlepnog algoritma na efikasan način moguće pronaći rešenje našeg problema.

# Implementacija algoritama

U nastavku ćemo razmotriti i predstaviti neke od poznatih algoritama za naš problem. Bitna svojstva svakog algoritma su potpunost i optimalnost. Potpunost je svojstvo koje garantuje da ce algoritam naći neko rešenje problema ako rešenja uopste postoje. Optimalno rešenje je rešenje sa najmanjom cenom i ono ne mora biti jednoznačno definisano. Moguće je da algoritam koji nema svojstvo optimalnosti često pronalazi rešenja bliska optimalnim, ali u značajno kraćem vremenu.

### Aproksimativni principalni postupak

Prvi algoritam koji analiziramo je osnovni pohlepni algoritam. Istraživanje nekog algoritma je bitno da bi se dati algoritam optimizovao i dobila optimalna rezolucija problema. Ukoliko se tim istrazivanjima pokaze da je problem NP-kompletan teško je dobiti algoritam polinomijalnog vremena za njegovo tačno rešavanje, ali ipak dok se ne dokaže suprotno verovatnoća za to i dalje postoji. Generalno, rešenje blizu optimalnog je obično dovoljno dobro. Principalni postupak rezultira skoro optimalnim rezultatom a poznat je i kao aproksimativni (Approk) princip. U nastavku je dat pseudokod.

1. C ← Ø

2. E′ ← E [G]

3. while E′ ≠ Ø

   4 neka je (u, v) proizvoljna ivica E′

   5 C ← C U {u, v}

   6 obrisati svaku ivicu iz E′ incidentnu sa u ili v

7. return C

Dati algoritam je naknadno implementiran u Python-u. U algoritmu dominira petlja koja bira

ivice iz datog grafa E i zatim uklanja sve grane koje su incidentne sa čvorovima izabrane grane, a datu granu smesta u skup resenja C. Vreme rada ovog algoritma je O (E).

**Algoritam pohlepna aproksimacija**

Da bismo pronašli rezultat najefikasnijeg ispitivanja, nalazimo grupu učesnika koja ima rezultat koji optimizuje (maksimizira ili minimizira) vrednost funkcije cilja. Postupak pohlepnog principa izvršava se korak po korak. Prvo su grupa odabranih učesnika prazni. Onda u svakom koraku pokušavamo pronaći i dodati najbolje učesnike od preostalih u grupu, tako da naš izbor mora biti vođen funkcijom izbora. Funkcija izbora zavisi od trenutnog ispitivanja. Na primer, funkcija izbora u slučaj sa stablom minimalne tezine selektuje ivicu minimalne težine od ostatka ivica, ili objekat sa maksimalnom tezinom od preostalih objekata ako je slučaj sa problemom rancem. Ako grupa odabranih učesnika više ne zadovoljava tačnost odbacujemo kandidata kojeg smo upravo dodali: uklonjen kandidat se nikada više ne razmatra. U projektima dakle, ako je veća grupa još uvek zadovoljiva nakon dodavanja kandidata, onda kandidat kojeg smo upravo dodali ostaje u grupi odabranih učesnika za sve vreme trajanja algoritma. Svaki put kada povećamo grupu odabranih učesnika, sledi ispitivanje da li grupa sada čine rešenje.

Opšte prihvaćena metoda za konstruisanje sukcesivnog prostora rezolucija je pohlepni pristup, tj. zasnovana je na dokazanom principu odabira (lokalnog) najboljeg izbora u svakoj fazi da bi se dobilo globalno najbolje resenje. Ispod teksta je dat pseudokod opsteg i pametnog pohlepnog algoritma.

*Opšti pohlepni algoritam*

1. C ← Ø

2. while E ≠ Ø

    3. Izabrati ivicu e ∈ E i izabrati čvor V ∈ e

    4. C ← C U {V}

    5. E ← E \ {e ∈ E : v ∈ e}

6. return C

*Pametni pohlepni algoritam*

1. C ← Ø

2. while E ≠ Ø

    3. Izabrati cvor v ∈ V maksimalnog stepena u trenutnom grafu

    4. C ← C U {v}

    5. E ← E \ {e ∈ E : v ∈ e}

6. return C

**Genetski algoritam**

Genetski algoritam je najefikasniji pristup zasnovan na prirodnoj selekciji. Održava niz hromozoma, poznatih kao nosače gena koji predstavljaju karakteristike problema. Princip postupka bira nekoliko nosioca roditeljskog gena iz grupa prema njihovoj sposobnosti, koje se analiziraju uz pomoć fitnes funkcije. Najprilagođeniji imaju veće šanse da budu izabrani za genetske operacije u budućnosti generacije. Primenjeni su različiti tipovi genetskih operatora nad roditeljima. U svakoj generaciji, novonastale jedinke se stvaraju pomoću delova najprilagođenijih roditelja stare generacije.

Iako genetski algoritam može ili ne mora dati ispravan rezultat, u većini slučajeva proizvodi bolju generaciju u poređenju sa roditeljskom generacijom. Kao sto su roditelji najprilagođeniji u generaciji i njihovi delovi opstaju u novim generacijama, tako i lošiji nosioci gena umiru u narednim generacijama. Ovaj postupak se ponavlja dok nije zadovoljen neki korisnički definisan završni kriterijum poznat kao funkcija cilja. Fitnes funkcija je obično, ali ne i nužno, jednaka funkciji cilja.

Genetski algoritmi pružaju varijaciju generacija inspirisani pridonom selekcijom i genetikom. Fitnes funkcija ima važnu ulogu u genetskim algoritmima, jer određuje koliko je dobar nosač gena. Fitnes funkcija može biti broj čvorova potrebnih za pokrivanje svih ivica u jednoj dimenziji.

$$M=\sum_{i=1}^{v} Vi \text{ where } V_i=1 \text{ if } V_i \in V_{cover} \text{ else } 0$$

Sledeća generacija se proizvodi od dva roditelja iz prethodne generacije, što predstavlja operator ukrštanja. Na taj način će 50% roditelja direktno otići u buduću generaciju uz pomoć reprodukcija. Svaki nosač gena je koristan za formiranje sledeće generacije pošto svaki hromozom ima nekoliko važnih gena koji mogu biti korisni za izbacivanje globalnog lokalnog minimuma. Tada se primenjuje još jedan genetski operator mutacija na sledeću generaciju. Mutacija predstavlja proces minimalne izmene genetskog sadržaja hromozoma i korisna je za izbacivanje lokalnog minimuma pa je zato treba primenjivati na svaku sledeću generaciju. U nastavku je dat pseudokod algoritma.

Pseudokod genetskog algortima

1. inicijalizuj populaciju

2. evaluiraj populaciju

3. while nije zadovoljen uslov za završetak

    4. odaberi roditelje za ukrštanje

    5. izvrši ukrštanje I mutaciju

    6. evaluiraj populaciju

7. end

# Analiza rezultata

U ovom delu biće isprobani i testirani gore navedeni algoritmi. Testiranje se vrši puštanjem u rad algoritama sa različitim veličinama ulaza. Utrošak vremena za svaki ulaz je meren pomoću funkcije biblioteke time, a grafici su iscrtani funkcijama biblioteke mathplotlib programskog jezika Python. Takođe, izvršeno je testiranje i za genetski algoritam sa različitim parametrima. Test podaci su tri grafa:

- GRAF1 - 5 cvorova, 10 grana,
- GRAF2 - 50 cvorova, 500 grana,
- GRAF3 - 1000cvorova, 25000 grana.

Za genetski algoritam korišćeni su sledeći parametri pre menjenja:

- populacija = 30,
- mutacija = 0.04,
- iteracije = 5,
- elitnaPopulacija = 8.

Rezultati izvršavanja različitih algoritama za ulaz GRAF1:

| Algoritam | Minimalni pokrivač |
|---|---|
| Aproksimativni | 2 |
| Pohlepni | 4 |
| Genetski | 4 |

Vreme izvršavanja različitih algoritama za ulaz GRAF2:

| Algoritam | Vreme (s) |
|---|---|
| Aproksimativni | 0.0001847 |
| Pohlepni | 0.0822159 |
| Genetski | 1.5232198 |

Rezultati izvršavanja različitih algoritama za ulaz GRAF3:

| Algoritam | Minimalni pokrivač |
|---|---|
| Aproksimativni | 499 |
| Pohlepni | 922 |
| Genetski | 928 |

Vreme izvršavanja aproksimativnog algoritma u zavisnosti od različitih ulaza:

| Ulaz | Vreme izvršavanja(s) |
|---|---|
| GRAF1 | 0.0001847 |
| GRAF2 | 0.0433030 |
| GRAF3 | 102.54411 |

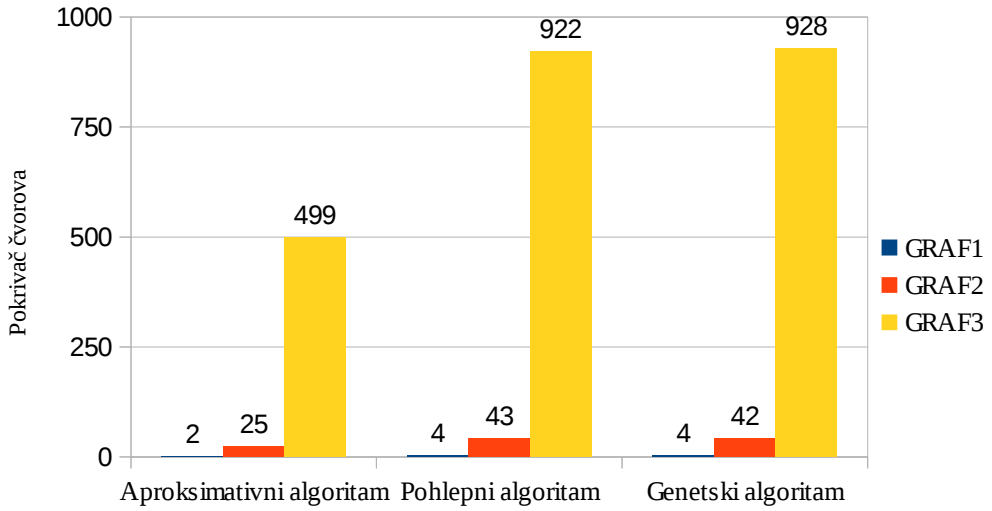
Vreme izvršavanja pohlepnog algoritma u zavisnosti od različitih ulaza:

| Ulaz | Vreme izvršavanja(s) |
|---|---|
| GRAF1 | 0.0007419 |
| GRAF2 | 0.0822159 |
| GRAF3 | 130.96070 |

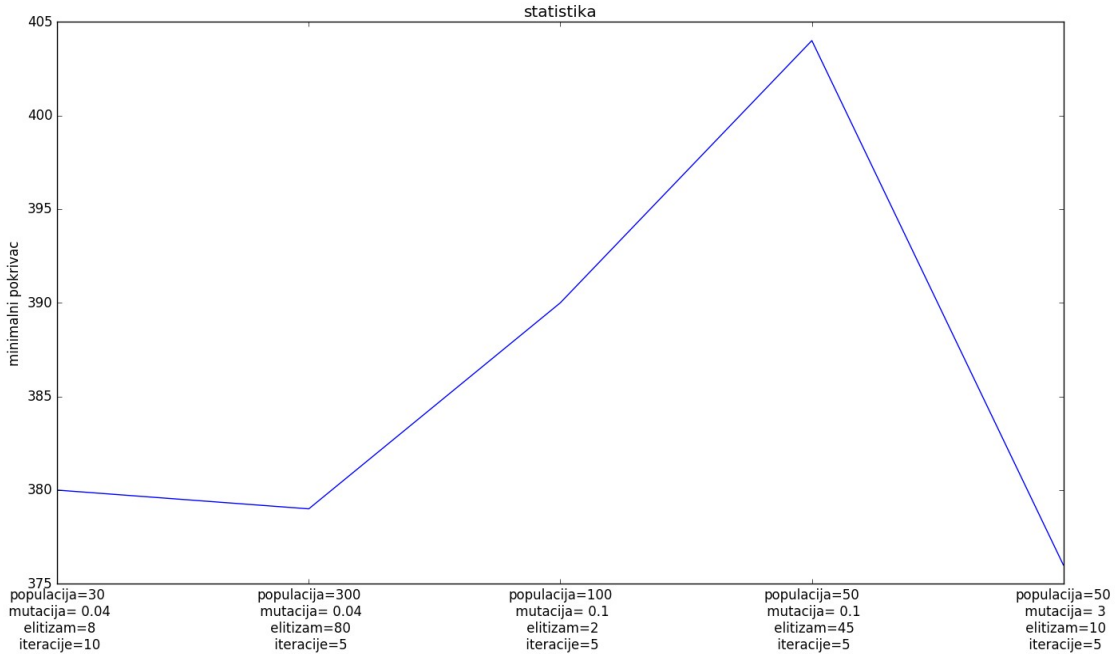
Vreme izvršavanja genetskog algoritma u zavisnosti od različitih ulaza:

| Ulaz | Vreme izvršavanja(s) |
|---|---|
| GRAF1 | 0.10293 |
| GRAF2 | 1.5232198 |
| GRAF3 | 139.553349 |

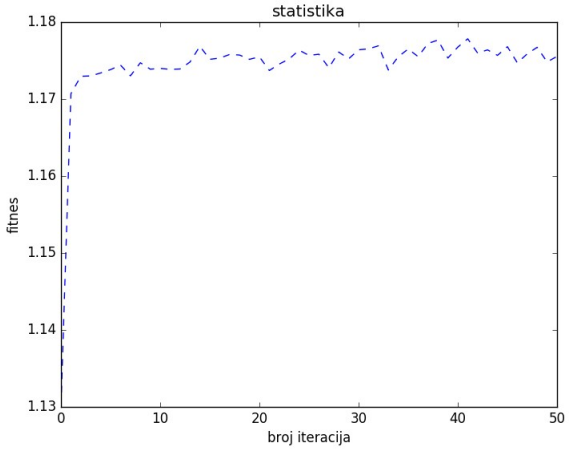
Rezultati izvršenih algoritama za sva tri ulazna grafa:



Rezultati izvršavanja genetskog algoritma u zavisnosti od različitih parametara za ulazni graf od 500 cvorova i 2500 grana:




Promena fitness funkcije tokom iteracija za GRAF3:




Složenost aproksimativnog algoritma je O(V+E). Ovo je dakle polinomijal algoritam koji ne garantuje da vraća najbolji rezultat ali je garantovano da ce dobijeno rešenje biti maksimalno dva puta lošije od najboljeg.Pohlepni pametni algoritam ima složenost O (logV) dok složenost genetskog zavisi od implementacije koja je korišćena.

### Još neki algoritmi iz literature

Pored implementiranih algoritama u literaturi se mogu pronaći jos mnogi sa različitim stepenom optimalnosti i slozenosti. Sada ćemo pomenuti neke.

Jedan od karakterističnih algoritama zbog svoje složenosti O(E) je algoritam koji je osmislio Monjural Alom. Ovaj algoritam daje bolje rešenje od postojećig aproksimativnog, pohlepnog i genetskog algoritma. Ova procedura bira čvor koji ima najveći stepen, I ako više od jednog čvora imaju maksimalni stepen onda se bira onaj čvor koji ima barem jednu granu koja nije pokrivena ostalim čvorovima koji imaju maksimalni stepen čvora grafa. Zatim se uklanjaju sve grane koje su incidente sa izabranim čvorom. Postupak se ponavlja dok nisu pokriveni svi čvorovi. Ovaj algoritam se izvršava duže kao što je potrebno vreme i aproksimativnom algoritmu ali uvek daje rešenje koje je bolje od aproksimatvnog. Važno je napomenuti da ovaj algoritam za veće grafove možda neće vratiti validno optimalno rešenje. Pošto je konstruisan tako da vraća svako moguce rešenje možemo ga modifikovati da od svih mogućih rešenja izaberemo jedno najbolje optimalno rešenje, ali je složenost u tom slučaju slična ostalim navedenim algoritmima.

Poznat je i takozvani Primal-dual postupak. Princip počinje sa nemogućom primarnom rezolucijom i izvodljivom dualnom. Sve dok traje izvršenje ove vrste principa poboljšava se dualnost ciljne funkcije, tj. imamo dvostruku rezoluciju takvu da se smanjuje stepen nemogućnosti primarnog i smanjuje stepen izvodljivosti dualnog dela u isto vreme. Princip se završava čim primarna rezolucija pocne da postoji. Konačna dulana rezolucija je kao donja granica za najbolju rezoluciju.Složenost ovakvog algoritma je O(V log V+E).

Bitno je pomenuti i algoritam Grana i granica(Branch and Bound).To je opšti princip za postizanje optimalnih rezultata za različite probleme u diskretnoj i kombinatornoj matematici.Sadrži sistematsko prikupljanje rezolucija svakog kandidata gde se beskorisni učesnici odbacuju, uz pomoć donje i gornje pretpostavljene granice količine optimizovanosti. Složenost ovoga algoritma medjutim eksponencijalno raste sa porastom velicine ulaza ali zato se garantuje tačan rezultat.

# Zaključak

Problem pokrivaca grafa može imati mnoge primene. U svakodnevnom životu ovim problemom mozemo modelovati situaciju gde je potrebno kamerama pokriti raskrsnice u gradu tako da su sve ulice vidljive. Ovde raskrsnicu mozemo smatrati čvorem a ulice granama. Naš problem je takodje prisutan i kod rešavanja mnogih kompleksnijih problema poput SVM sklapanja u biohemiji ili sigurnosti kompjuterskih I mobilnih mreza.

Što se tice aproksimacije datog algoritma naučno je dokazano da je minimalna aproksimacija maksimum 1.1666, a trenutni rezultati pokazuju aproksimaciju sa

$$2 - \frac{\log \log |V|}{2 \log |V|}$$

# Literatura

[1]U. Manber, Introduction to algorithms, A creative approach, Addison–Wesley, Reading, 1989.

[2] Miodrag Živković, Algoritmi, Matematički fakultet,Beograd

[3]Combinatorial Optimization: Algorithms and Complexity, Christos H. Papadimitriou, Kenneth Steiglitz

[4]Applications of Graph Theory,Ashay Dharwadker and Shariefuddin Pirzada, JOURNAL OF THE KOREAN SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS VOL. 11 NO. 4 2007

[5] Soumya Godi et al. —Several Algorithms to solve vertex Cover Problem‖ IJCMS, Vol.4, Issue 4, April 2015

[6] Mohammed Eshtey et al. —NMVSA greedy solution for Vertex Cover Problem‖ IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 3, 2016