UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

DIPLOMSKE AKADEMSKE STUDIJE-MASTER

MASTER RAD

Algoritmi za pronalaženje maksimalnih klika u grafu

Mentor: Prof.dr Miodrag Živković

Student: Snježana Jović
1026/2008

Beograd, Januar 2012.g.

SADRŽAJ

1. Uvod

Termin „klika“ i problem algoritamskog ispisivanja klika potiču iz društvenih nauka. Problem klike je problem pronalaženja, unutar društvene mreže, najveće grupe ljudi, u kojoj se svi međusobno poznaju.

U vezi sa klikama u grafu moguće je razmatrati više problema:

- Pronalaženje maksimalne klike
- Ispisivanje svih maksimalnih klika
- Pronalaženje maksimalne težinske klike u težinskom grafu
- Ispitivanje da li graf sadrži kliku date veličine

U ovom radu se razmatraju prva dva problema tj. rad se bavi pronalaženjem i ispisivanjem svih maksimalnih klika u datom neusmjerenom grafu.

Termin „klika“ u vezi sa grafovima uveli su Lus (Luce) i Peri (Perry). Autori prvog algoritma za pronalaženje svih maksimalnih klika u grafu su Harari (Harary) i Ros (Ross) (1957). Otada, pronađeno je mnogo algoritama za različite verzije problema klike. Počevši od rada Kuka (Cook) i Karpa (Karp) iz 1972. godine dokazano je da je problem klika NP kompletan. Mun (Moon) i Mozer (Moser) 1965. godine su dokazali da u grafu sa n čvorova ima najviše $3^{\frac{n}{3}}$ maksimalnih klika. Najstariji algoritam za ispisivanje svih maksimalnih klika u grafu je algoritam Brona (Bron) i Kerboša (Kerbosh) (1973) [3,6] koji koristi pretragu sa vraćanjem (backtracking). Za neke varijante Bron-Kerboš algoritma pokazano je da im je složenost u najgorem slučaju $O(3^{\frac{n}{3}})$. Algoritam Karagana (Carraghan) i Pardalosa (Pardalos) (1990) [2] koristi metodu grananja i ograničavanja (branch and bound), kao i algoritam Estergarda (Östergärd) (2002) [9], odnosno algoritam Cliquer (2003) [8].

Cukijama (Tsukiyama) je pokazao da je moguće ispisati sve maksimalne klike u grafu tako da je prosječno vrijeme po jednoj ispisanoj kliki polinomijalno. Ako se zna da u grafu nema eksponencijalan broj klika, tada se problem klike može riješiti u polinomijalnom vremenu. Najbrži poznati algoritam za rješavanje problema klike je algoritam Robsona. Njogova složenost je $O(2^{0.276n}) = O(1.2108^n)$.

Pored primjene u društvenim naukama, problem klike takođe ima značajne primjene u bioinformatici i računarskoj hemiji.

Rad se sastoji od šest poglavlja. Prvo poglavlje predstavlja uvod. Drugo poglavlje sadrži definicije neophodnih pojmova. U trećem poglavlju su opisana četiri algoritma za pronalaženje maksimalnih klika u neusmjerenom grafu. To su algoritam Brona i Kerboša, algoritam Karagana i Pardalosa, algoritam Estergarda i algoritam iz programa Cliquer. Opisane su dvije heuristike za izbor redoslijeda čvorova u grafu. Takođe je izvršeno upoređivanje pomenutih algoritama. U četvrtom poglavlju opisana je programska realizacija prethodno pomenutih algoritama. Peto poglavlje sadrži opis benčmark grafova i rezultate

izvršavanja programa za te grafove sa i bez primjenjenih heuristika za redoslijed čvorova. Šesto poglavlje sadrži zaključak i moguće pravce daljeg rada.

2. Osnovni pojmovi

U ovom poglavlju su opisani osnovni pojmovi koji će biti korišćeni u radu.

Definicija 1: Graf je uređeni par $G = (V, E)$, gde je V konačan skup čvorova, a $E \subset V \times V$ je skup grana. Neka je $V = \{v_1, v_2, \dots, v_n\}$ tada je $E = \{(v_i, v_j) | i \neq j, v_i, v_j \in V\}$. Čvorovi v_i, v_j su **povezani** ako $(v_i, v_j) \in E$.

U grafovima koji će biti razmatrani u ovom radu važi $(v_i, v_j) \in E$, $v_i \neq v_j$, $v_i, v_j \in V$.

Definicija 2 : $G = (V, E)$ je **neusmjeren graf** ako i samo ako je za svako $v_i, v_j \in V$, iz $(v_i, v_j) \in E$ slijedi $(v_j, v_i) \in E$. Grane (v_i, v_j) i (v_j, v_i) predstavljaju istu granu.

Definicija 3: Neka je $G = (V, E)$ proizvoljan neusmjereni graf, gdje je $V = \{v_1, v_2, \dots, v_n\}$ skup čvorova grafa G i $E \subseteq V \times V$ skup njegovih grana. Broj čvorova sa kojima je povezan čvor $v \in V$ naziva se **stepen čvora** v i označava se sa $d(v)$. Važi da je $d(v) < n$.

Definicija 4: Komplement grafa G je graf $\bar{G} = (V, \bar{E})$, gdje je $\bar{E} = \{(v_i, v_j) | v_i, v_j \in V, (v_i, v_j) \notin E\} \setminus \{(v_i, v_i) | i \in \{1 \dots n\}\}$

Definicija 5: Graf $G = (V, E)$ je **potpuni graf** ako je svaki čvor iz V povezan sa svim ostalim čvorovima iz V.

Definicija 6: Graf $G' = (V', E')$ je **podgraf** grafa $G = (V, E)$ ako i samo ako je $V' \subseteq V$ i $E' \subseteq E$,tako da iz $(v_i, v_j) \in E'$ slijedi $v_i, v_j \in V'$.

Definicija 7: Graf $G' = (V', E')$ je **indukovani podgraf** grafa $G = (V, E)$ ako je $V' \subseteq V$ i E' sadrži sve grane iz E kojima su oba kraja u V'. Pošto skup V' jednoznačno određuje indukovani podgraf, može se reći da je V' indukovani podgraf.

Definicija 8: Klika u G je indukovani podgraf $S \subseteq V$, čija su svaka dva čvora susjedna (odnosno indukovani potpuni podgraf). **Veličina klike** je broj čvorova koje klika sadrži. Za kliku kažemo da je **neproširiva** ako ona nije podskup bilo koje veće klike, te da je **maksimalna** ako ne postoji veća klika u grafu.

Definicija 9: Bojenje grafa G je preslikavanje $\beta: V \to B$ čvorova u skup boja B, tako da iz $(i, j) \in E$ slijedi $\beta(i) \neq \beta(j)$.Drugim riječima svakom čvoru pridružena je neka boja, a susjednim čvorovima uvijek su pridružene različite boje.

Neka je dat graf $G = (V, E)$ i broj k. Poznato je da je problem utvrđivanja da li u G postoji klika veličine k (problem klika) NP kompletan [13]. Zbog toga je problem pronalaženja svih maksimalnih klika u datom grafu težak.

Definicija 10: Nezavisan skup je skup čvorova $V' \subset V$ grafa G koji nisu povezani ni sa jednim drugim čvorom iz V'. Problem maksimalnog nezavisnog skupa je pronalaženje nezavisnog skupa maksimalne kardinalnosti.

Definicija 11: Pokrivač grana grafa G je skup $V' \subset V$ takav da za svaku granu $(i, j) \in E$ važi da je bar jedan od čvorova i, j u skupu V'. Problem minimalnog pokrivača grana je problem pronalaženja pokrivača minimalne kardinalnosti.

Lako je vidjeti da su sljedeća tvrđenja ekvivalentna:

skup S je klika u grafu G

S je nezavisan skup u grafu $\bar{G}$

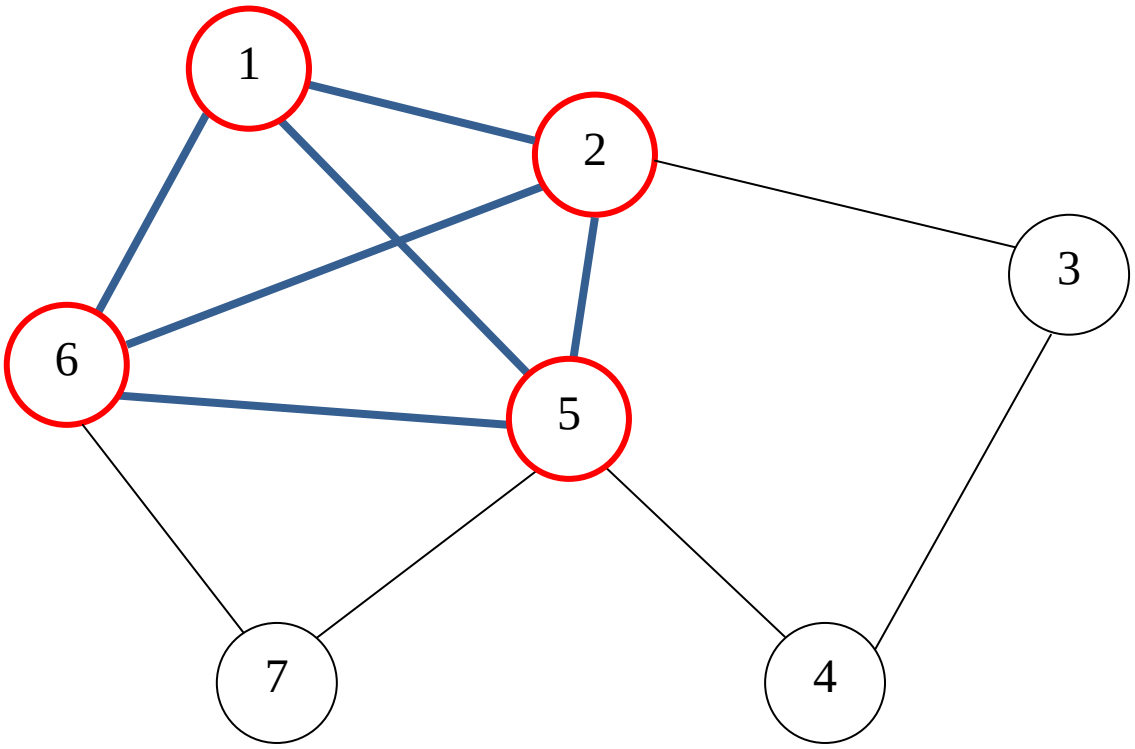
skup $V\backslash S$ je pokrivač grana u grafu $\bar{G}$.

Zbog toga bilo koji rezultat koji se odnosi na jedan od ovih pojmova ima svoj ekvivalentni oblik za ostala dva pojma.

Definicija 12: Graf se može predstaviti pomoću **matrice povezanosti**. To je simetrična $n\times n$ matrica $A_G = (a_{ij})_{(v_i,v_j)\in V\times V}$, gdje je $a_{ij} = 1$ ako je $(v_i, v_j) \in E$ jedna grana od G, i $a_{ij} = 0$ ako $(v_i, v_j) \notin E$.

Primjer 1.

Na Slici 1. je prikazan primer graf sa 7 čvorova i 11 grana. U ovom grafu su klike na primjer skupovi {5,6,7}, {1,2,5,6}. Skup {2,3,4,5} nije klika zato što čvorovi 2 i 4 nisu povezani. Skup {1,2,5,6} je maksimalna klika.



Slika 1. Primjer grafa sa sedam čvorova i maksimalnom klikom veličine 4.

Matrica povezanosti ovog grafa je:

0 1 0 0 1 1 0
1 0 1 0 1 1 0
0 1 0 1 0 0 0
0 0 1 0 1 0 0
1 1 0 1 0 1 1
1 1 0 0 1 0 1
0 0 0 0 1 1 0

3. Algoritmi za pronalaženje maksimalnih klika

U ovom poglavlju razmatra se algoritam Brona i Kerboša za pronalaženje svih neproširivih klika, kao i tri algoritma za pronalaženje maksimalnih klika u datom neusmjerenom grafu. To su algoritam Brona i Kerboša, algoritam Karagana i Pardalosa, algoritam Estergarda i algoritam iz programa Cliquer. U tački 3.5. su izvršena upoređivanja ovih algoritama i uočene su njihove sličnosti i razlike.

Izdvajamo pojmove koji su korišćeni u svim pomenutim algoritmima.

Neka je dat neusmjereni graf $G = (V, E)$.

Neka je $n = |V|, V = \{w_1, w_2, \dots, w_n\}$ skup čvorova grafa,

$N(w)$- skup čvorova koji su susjedni sa čvorom w.

U daljem tekstu kada se kaže da je čvor w_i veći od čvora w_j to znači da je indeks čvora w_i veći od indeksa čvora w_j.

3.1. Algoritam Brona i Kerboša

Ovaj algoritam [2,3] je jedan od najstarijih algoritama koji pronalaze i ispisuju sve neproširive klike. Osnovna verzija algoritma Brona i Kerboša, koja je opisana u ovom radu, je rekurzivni algoritam pretrage, koji pronalazi sve neproširive klike u neusmjerenom grafu $G = (V, E)$. Algoritam se lako može prepraviti tako da ispisuje sve maksimalne klike u neusmjerenom grafu.

Algoritam $BKklika(K, U, S)$**//svi čvorovi u skupu** U **su veći od svih čvorova u skupu** K
// svi čvorovi iz skupa S **su manji od najvećeg čvora u skupu** K

```
1:    if U == ∅ and S == ∅ then
2:        štampati K;
3:    else
4:        U = {w_1, w_2, ..., w_p} // w_1 < w_2 < ⋯ < w_p
5:        for i = 1 to p do
6:            U = U\{w_i};
7:            BKklika(K ∪ {w_i}, U ∩ N(w_i), S ∩ N(w_i));
8:            S = S ∪ {w_i};
Program glavni;
9:    BKklika(∅, V, ∅);
10:   return;
```

Slika2. Algoritam Brona i Kerboša

Neka je fiksiran redoslijed čvorova $\{w_1, w_2, \dots, w_n\}$ u zadatom grafu G i neka za čvorove važi $w_1 < w_2 < \cdots < w_n$. Neka klika K sadrži j ($j \geq 0$) čvorova sa indeksima $i_1 < i_2 < \cdots < i_j$ i neka je U skup čvorova iz G sa indeksima većim od i_j koji su granama povezani sa svim čvorovima iz K (za $j = 0, U$ sadrži svih n čvorova). Neka je S skup svih čvorova koji ne mogu dalje biti korišćeni za dopunjavanje skupa K (jer su svi čvorovi iz S manji od najvećeg

čvora iz K, pa su maksimalne klike koje sadrže čvorove iz S već pronađene), a povezani su sa svim čvorovima iz K. Za svaki izabrani čvor $w_i \in U$, u petlji se proširuje skup K i dobija se $K \cup \{w_i\}$, a onda se vrši rekurzivni poziv čvorovima iz skupa U', koji se od U dobija izbacivanjem čvora w_i i čvorova koji nisu susedni sa w_i, kao i svih čvorova sa indeksima manjim od i. Takođe, u rekurzivnom pozivu se umjesto S koristi skup S', koji se dobija od S izbacivanjem svih čvorova koji nisu susjedni sa w_i. Kada su skupovi U i S prazni, dobijena klika je neproširiva i ispisuje se. Izloženi algoritam je opisan kodom na slici 2.

Na početku skupovi K i S su prazni, a skup U sadrži sve čvorove grafa G. Važna činjenica je da se svi čvorovi koji su povezani sa svim čvorovima u K nalaze ili u skupu U ili u skupu S. Čvorovi iz skupa U se koriste za proširivanje skupa K. Čvor koji se jednom iskoristi za proširivanje skupa K, premješta se u skup S, zato što u skup K ulazi čvor sa većim indeksom od njegovog. Skup S sadrži čvorove koji su prethodno bili u U i čiji su indeksi svi manji od najvećeg indeksa u K, pa su sve neproširive klike koje ih sadrže već ispisane. Svrha održavanja skupa S je da se izbjegne ispisivanje istih neproširivih klika više puta. Da bi se izbjeglo ispisivanje klika koje su manje od do sada nađene maksimalne klike, algoritam provjerava da li je skup S prazan. Ako S nije prazan, čvorovi iz skupa S bi mogli biti dodati skupu K, ali ovo bi dalo prethodno pronađenu maksimalnu kliku. Zbog načina na koji nastaje skup K, očigledno je da njegovi elementi predstavljaju kliku. Zbog pretpostavke da su polazni čvorovi uređeni u rastućem poretku prema indeksima čvorova, klike se generišu u leksikografskom poretku.
Dokaz korektnosti zasniva se na sljedećem tvrđenju.

Teorema 1. U toku izvršavanja algoritma $BKklika(K, U, S)$

(i) Svi čvorovi u skupu U su veći od svih čvorova u skupu K.

(ii) Svi čvorovi iz skupa S su manji od najvećeg čvora u skupu K.

Dokaz:
Dokaz se izvodi principom matematičke indukcije.

Baza indukcije: Na početku algoritma skupovi K i S su prazni, a skup U sadrži sve čvorove grafa G. Prema tome važi (i) jer je skup K prazan, a važi i (ii) jer je skup S prazan.

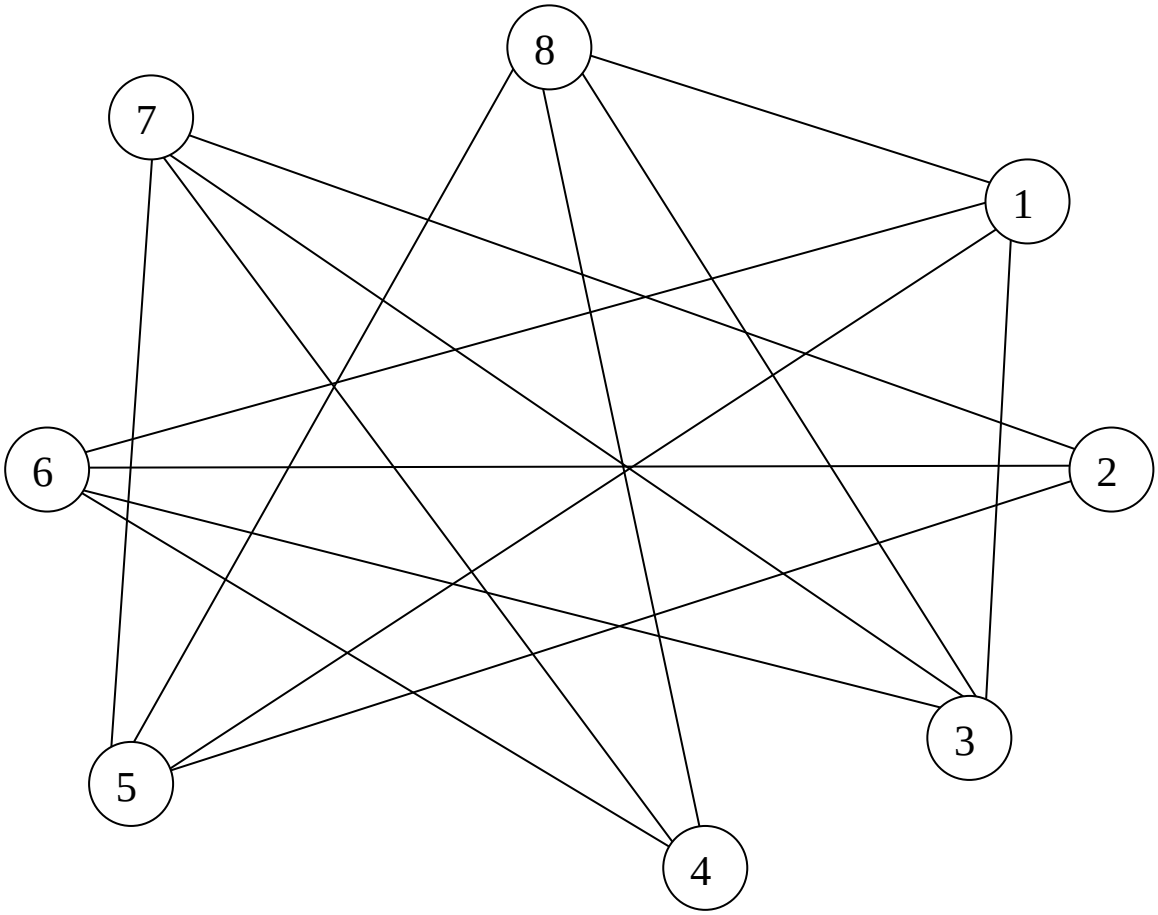
Indukcijska hipoteza: Pretpostavimo da tvrđenja (i) i (ii) važe za poziv $BKklika(K, U, S)$.

Korak indukcije: Dokažimo da tvrđenje važi za poziv $BKklika(K \cup \{w_i\}, U \cap N(w_i), S \cap N(w_i))$.

Po pretpostavci važi da su svi čvorovi u skupu U veći od svih čvorova u skupu K. U liniji 6 bira se čvor w_i sa najmanjim indeksom u skupu U. Taj čvor se izbacuje iz skupa U i dodaje se skupu K. Time su i dalje svi čvorovi iz skupa U veći od svih čvorova iz skupa K. Zatim se iz skupa U izbace svi čvorovi koji nisu povezani sa w_i , ali i dalje svi čvorovi iz skupa U su veći od svih čvorova iz skupa K. Time je dokazano da tvrđenje (i) važi za svaki poziv $BKklika(K, U, S)$.

Po pretpostavci za poziv $BKklika(K, U, S)$ važi da su svi čvorovi iz skupa S manji od najvećeg čvora u skupu K. Skup $K \cup \{w_i\}$ predstavlja skup u kome je w_i čvor sa najvećim indeksom, a $S \cap N(w_i)$ se dobija od skupa S kada se iz njega izbace svi čvorovi koji nisu povezani sa w_i tako da na osnovu pretpostavke važi da su svi čvorovi iz skupa $S \cap N(w_i)$

manji od najvećeg čvora u $K \cup \{w_i\}$. Time je dokazano da tvrđenje (ii) važi za svaki poziv $BKklika(K \cup \{w_i\}, U \cap N(w_i), S \cap N(w_i))$.



Slika 3.Graf za ilustraciju algoritama za pronalaženje maksimalnih klika.

Primjer 1. Razmotrimo graf G sa skupom čvorova $\{1,2,\dots,8\}$ prikazan na Slici 3. Primjena na ovaj graf algoritma $BKklika$, koji je prilagođen da ispisuje samo maksimalne klike, opisuje se nizom koraka u nastavku. Pri tome je dubina rekurzivnog poziva predstavljena dubinom uvlačenja odgovarajućeg reda.

1: $K = \{\}; U = \{1,2,3,4,5,6,7,8\}; S = \{\};$
2: $K = \{1\}; U = \{3,5,6,8\}; S = \{\};$
    3: $K = \{1,3\}; U = \{6,8\}; S = \{\};$
        4: $K = \{1,3,6\}; U = \{\}; S = \{\};$ max $= 3$ Pronađena je klika $\{1,3,6\}$;
        5: $S = \{6\};$
        6: $K = \{1,3,8\}; U = \{\}; S = \{\};$ max $= 3$ Nova klika je $\{1,3,8\}$;
        7: $S = \{6,8\};$
    8: $S = \{3\};$
    9: $K = \{1,5\}; U = \{8\}; S = \{\};$
        10: $K = \{1,5,8\}; U = \{\}; S = \{\};$ max $= 3$ Nova klika je $\{1,5,8\}$;
        11: $S = \{8\};$

12: $S = \{3,5\}$;
13: $K = \{1,6\}; U = \{\}; S = \{3\}$;
14: $S = \{3,5,6\}$;
15: $K = \{1,8\}; U = \{\}; S = \{3,5\}$;
16: $S = \{3,5,6,8\}$;
17: $S = \{1\}$;
18: $K = \{2\}; U = \{5,6,7\}; S = \{\}$;
19: $K = \{2,5\}; U = \{7\}; S = \{\}$;
20: $K = \{2,5,7\}; U = \{\}; S = \{\};$ max $= 3$ Nova klika je $\{2,5,7\}$;
21: $S = \{7\}$;
22: $S = \{5\}$;
23: $K = \{2,6\}; U = \{\}; S = \{\}$;
24: $S = \{5,6\}$;
25: $K = \{2,7\}; U = \{\}; S = \{5\}$;
26: $S = \{5,6,7\}$;
27: $S = \{1,2\}$;
28: $K = \{3\}; U = \{6,7,8\}; S = \{1\}$;
29: $K = \{3,6\}; U = \{\}; S = \{1\}$;
30: $S = \{1,6\}$;
31: $K = \{3,7\}; U = \{\}; S = \{\}$;
32: $S = \{1,6,7\}$;
33: $K = \{3,8\}; U = \{\}; S = \{1\}$;
34: $S = \{1,6,7,8\}$;
35: $S = \{1,2,3\}$;
36: $K = \{4\}; U = \{6,7,8\}; S = \{\}$;
37: $K = \{4,6\}; U = \{\}; S = \{\}$;
38: $S = \{6\}$;
39: $K = \{4,7\}; U = \{\}; S = \{\}$;
40: $S = \{6,7\}$;
41: $K = \{4,8\}; U = \{\}; S = \{\}$;
42: $S = \{6,7,8\}$;
43: $S = \{1,2,3,4\}$;
44: $K = \{5\}; U = \{7,8\}; S = \{1,2\}$;
45: $K = \{5,7\}; U = \{\}; S = \{2\}$;
46: $S = \{1,2,7\}$;
47: $K = \{5,8\}; U = \{\}; S = \{1\}$;
48: $S = \{1,2,7,8\}$;
49: $S = \{1,2,3,4,5\}$;
50: $K = \{6\}; U = \{\}; S = \{1,2,3,4\}$;
51: $S = \{1,2,3,4,5,6\}$;
52: $K = \{7\}; U = \{\}; S = \{2,3,4,5\}$;
53: $S = \{1,2,3,4,5,6,7\}$;
54: $K = \{8\}; U = \{\}; S = \{1,3,4,5\}$;
55: $S = \{1,2,3,4,5,6,7,8\}$;

Pokazalo se da ova verzija Bron Kerboš algoritma nije efikasna za grafove sa mnogo neproširivih klika koje nisu maksimalne, jer pravi rekurzivni poziv za svaku kliku bilo da je ona maksimalna ili ne.

3.2. Algoritam Karagana i Pardalosa

U svom radu [9] Estergard kao predstavnika starih algoritama za pronalaženje svih maksimalnih klika navodi algoritam Karagana i Pardalosa [2]. Ovaj algoritam je sličan rekurzivnom algoritmu za generisanje leksikografskim redosledom svih podskupova veličine k (kombinacija) skupa $N = \{1,2,\dots,n\}$. Familija traženih podskupova koji sadrže j fiksiranih elemenata $i_1 < i_2 < \cdots < i_j$ skupa N može se rekurzivno proširiti do familija $i_1, i_2, \dots, i_j, r$ koje sadrže $j+1$ elemenata dodavanjem elementa r, $r = i_j + 1, i_j + 2, \dots, n-k+j+1$, $j = 0,1,\dots,k$ (naravno, za $j = k$ ne ulazi se u rekurziju). Kada se generišu klike, dodaje se provjera povezanosti čvorova u podskupu.

Izloženi algoritam je prikazan na slici 4.

```
Algoritam KPklika(K, U)// svi čvorovi u skupu U su veći od svih čvorova u skupu K
 1:   if U == Ø then
 2:       if |K| ≥ max then
 3:           max = |K|;
 4:           štampati K;
 5:       return
 6:   while U ≠ Ø do        // čvorovi iz K čine kliku povezanu sa svim čvorovima iz U
 7:       if |K ∪ U| < max then          //odsijecanje
 8:           return
 9:       i = najmanji redni broj nekog čvora iz skupa U;
10:       U = U\{w_i};                   //smanjivanje skupa U
11:       KPklika(K ∪ {w_i}, U ∩ N(w_i));   //rekurzivni poziv za K ∪ {w_i}
12:   return;
Program glavni;
13:   max = 0;
14:   KPklika(Ø, V);
15:   return;
```

Slika 4. Algorita Karagana i Pardalosa

Neka je fiksiran redoslijed čvorova $\{w_1, w_2, \dots, w_n\}$ u zadatom grafu G i neka za čvorove važi $w_1 < w_2 < \cdots < w_n$. Neka je max globalna promjenljiva čija je vrednost veličina najveće trenutno pronađene klike u G. Neka klika K sadrži j ($j \geq 0$) čvorova sa indeksima $i_1 < i_2 < \cdots < i_j$ i neka je U skup čvorova iz G sa indeksima većim od i_j koji su granama povezani sa svim čvorovima iz K (za $j = 0$ U sadrži svih n čvorova). Naredni čvor u kliki koja sadrži K može biti bilo koji čvor $w_i \in U$. Za svaki izabrani čvor $w_i \in U$, se u petlji klika $K' = K \cup \{w_i\}$ rekurzivno proširuje čvorovima iz skupa $U' = U \cap N(w_i)$, koji se od U dobija izbacivanjem svih čvorova sa indeksima manjim od i i čvorova koji nisu susedni sa w_i. Naravno, ako je $U = \emptyset$, ne ulazi se u rekurziju; ako je u tom trenutku $j > max$, to znači da je pronađena klika veća od prethodno najveće, pa se max zamjenjuje sa j. Činjenica da je veličina klike koja sadrži K ograničena sa $|K \cup U|$ omogućuje odsijecanje (tj. postaje nepotrebno pregledanje klika koje sadrže K kao svoj podskup) ako je $|K \cup U| < max$. Posle završetka pretrage max je jednako veličini maksimalne klike u G.

Dokaz korektnosti zasniva se na sljedećem tvrđenju.

Teorema 2. U toku izvršavanja algoritma $KPklika(K, U)$ svi čvorovi u skupu U su veći od svih čvorova u skupu K. Svi čvorovi iz U povezani su sa svim čvorovima iz K.
Dokaz se izvodi principom matematičke indukcije.

Baza indukcije: Na početku algoritma skup K je prazan, a skup U sadrži sve čvorove grafa G. Prema tome tvrđenje važi jer je skup K prazan.

Indukcijska hipoteza: Pretpostavimo da tvrđenje važi za poziv $KPklika(K, U)$.

Korak indukcije: Dokažimo da tvrđenje važi za poziv $KPklika\big(K \cup \{w_i\}, U \cap N(w_i)\big)$. Po pretpostavci važi da su svi čvorovi u skupu U veći od svih čvorova u skupu K. U liniji 9 bira se čvor w_i sa najmanjim indeksom u skupu U. Taj čvor se izbacuje iz skupa U i dodaje se skupu K. Time su i dalje svi čvorovi iz skupa U veći od svih čvorova iz skupa K. Zatim se iz skupa U izbace svi čvorovi koji nisu povezani sa w_i, ali i dalje svi čvorovi iz skupa U su veći od svih čvorova iz skupa K. Jasno je da su svi čvorovi iz U povezani sa svim čvorovima iz K, zbog načina na koji nastaje skup K, što znači da čvorovi skupa K čine kliku. Time je dokazano da tvrđenje važi za svaki poziv $KPklika\big(K \cup \{w_i\}, U \cap N(w_i)\big)$.

Algoritam $KPklika$ se razlikuje od algoritma $BKklika$ po tome što koristi odsjecanje, koje je prikazano u liniji 7. Ako važi da je $|K \cup U| < max$, tada je pronalaženje klike veličine max u skupu $K \cup U$ nemoguće, pa se izvršava odsijecanje. Skup U je skup čvorova kojima se može proširiti trenutna klika. Ako je zbir broja čvorova koji čine trenutnu kliku i čvorova koji se mogu dodati na tu kliku manji od veličine trenutne pronađene maksimalne klike, tada dalje pretraživanje skupa U nema smisla, jer se ne može pronaći nova maksimalna klika i vrši se odsijecanje pretrage.

Efikasnost algoritma zavisi od izabranog redoslijeda čvorova. Pitanje heuristika za izbor redoslijeda čvorova razmatraćemo u tački 3.6.

Bez odsijecanja u liniji 7, algoritam bi pregledao svaku kliku u grafu. Ako se traže klike unaprijed zadate veličine, ovo odsijecanje postaje aktivno od samog početka pretrage.

Iako je ovaj algoritam jednostavan, prema [6] on je trenutno najbolji algoritam za rijetke grafove (grafove sa malim brojem grana).

Većina poboljšanja ovog algoritma zasnivaju se na metodima za ocjenu gornje granice veličine očekivane klike (preciznijim od $|K \cup U|$). Ocjena se obično zasniva na nekom bojenju čvorova grafa, takvom da susjedni čvorovi moraju biti obojeni različitim bojama. Ako se neki graf (ili indukovani podgraf) može obojiti sa s boja, onda on ne može da sadrži kliku veću od $s + 1$. Prilikom izbora strategija korišćenja gornjih granica potrebno je napraviti kompromis: bojenje može da znatno smanji broj čvorova u stablu pretrage, ali može da troši mnogo vremena.

Primjer 2. Razmotrimo graf G sa skupom čvorova $\{1,2,\dots,8\}$ prikazan na Slici 3. Primjena algoritma $KPklika$ na ovaj graf opisuje se nizom koraka u nastavku. Pri tome je dubina rekurzivnog poziva predstavljena dubinom uvlačenja odgovarajućeg reda, kao u Primjeru 1.

1: $K = \{\}$; $U = \{1,2,3,4,5,6,7,8\}$; $|K \cup U| = 8$;
2: $K = \{1\}$; $U = \{3,5,6,8\}$; $|K \cup U| = 5$;
3: $K = \{1,3\}$; $U = \{6,8\}$; $|K \cup U| = 4$;
4: $U = \{\}$; $max = 3$ Pronadjena je klika $\{1,3,6\}$;
5: $K = \{1,3\}$; $U = \{8\}$; $|K \cup U| = 3$;
6: $U = \{\}$; $max = 3$ Nova klika je $\{1,3,8\}$;
7:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;
8: $K = \{1\}$; $U = \{5,6,8\}$; $|K \cup U| = 4$;
9: $K = \{1,5\}$; $U = \{8\}$; $|K \cup U| = 3$;
10: $U = \{\}$; $max = 3$ Nova klika je $\{1,5,8\}$;
11:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;
12: $K = \{1\}$; $U = \{6,8\}$; $|K \cup U| = 3$;
13: $U = \{\}$; $K < max$;
14:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;
15: $K = \{\}; U = \{2,3,4,5,6,7,8\}$; $|K \cup U| = 7$;
16: $K = \{2\}$; $U = \{5,6,7\}$; $|K \cup U| = 4$;
17: $K = \{2,5\}$; $U = \{7\}$; $|K \cup U| = 3$;
18: $U = \{\}$; $max = 3$ Nova klika je $\{2,5,7\}$;
19:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;
20: $K = \{2\}$; $U = \{6,7\}$; $|K \cup U| = 3$;
21: $U = \{\}$; $K < max$;
22:Odsijecanje $(|K \cup U| < max)\ |K \cup U| = 2$
23: $K = \{\}$; $U = \{3,4,5,6,7,8\}$; $|K \cup U| = 6$;
24: $K = \{3\}$; $U = \{6,7,8\}$; $|K\ U\ U| = 4$;
25: $U = \{\}$; $K < max$
26: $K = \{3\}$; $U = \{7,8\}$; $|K\ U\ U| = 3$;
27: $U = \{\}$; $K < max$;
28:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;
29: $K = \{\}$; $U = \{4,5,6,7,8\}$; $|K\ U\ U| = 5$;
30: $K = \{4\}$; $U = \{6,7,8\}$; $|K \cup U| = 4$;
31: $U = \{\}$; $K < max$;
32: $K = \{4\}$; $U = \{7,8\}$; $|K \cup U| = 3$;
33: $U = \{\}$; $K < max$;
34:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;
35: $K = \{\}$; $U = \{5,6,7,8\}$; $|K \cup U| = 4$;
36: $K = \{5\}$; $U = \{7,8\}$; $|K\ U\ U| = 3$;
37: $U = \{\}$; $K < max$
38:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;
39: $K = \{\}$; $U = \{6,7,8\}$; $|K \cup U| = 3$;
40: $U = \{\}$; $K < max$;
41:Odsijecanje $(|K \cup U| < max)|K \cup U| = 2$;

3.3. Algoritam Estergarda

U ovom potpoglavlju predstavljen je algoritam baziran na ideji korišćenja nezavisnih skupova ili klasa boja. On takođe koristi pretragu sa vraćanjem [9]. Novi algoritam je baziran na ideji grananja i ograničavanja [19] i predstavlja razvijeniju verziju algoritma Karagana i Pardalosa [2], koji je pokazao svoju efikasnost u mnogim testovima.

Neka je fiksiran redoslijed čvorova$\{w_1, w_2, \dots, w_n\}$ u zadatom grafu G i neka je $S_i = \{w_i, w_{i+1}, \dots, w_n\},\ i = 1, \dots, n$. Algoritam Estergarda traži klike koje sadrže čvor w_i u skupovima S_i redom za $i = n, n-1, \dots, 1$. Svako od tih traženja odvija se kao u algoritmu Karagana i Pardalosa. Ovakva organizacija algoritma omogućuje skraćivanje pretrage. Algoritam Estergarda je prikazan na slici 5.

```
Algoritam Eklika( K, U)// svi čvorovi u skupu U su veći od svih čvorova u skupu K
1:    if U == ∅ then
2:        if |K| ≥ max then
3:            max = |K|;
4:            štampati K;
5:        return
6:    while U ≠ ∅ do
7:        if |K ∪ U| < max then            // odsijecanje 1
8:            return
9:        i = najmanji redni broj nekog čvora iz skupa U;
10:       if |K| + c[i] < max then         // odsijecanje 2
11:           return
12:       U = U\{w_i} ;
13:       Eklika(K ∪ {w_i}, U ∩ N(w_i))    // čvor w_i uključen je u kliku
14:   return
Program glavni;
15:   max = 0;
16:   for i = n downto 1 do
17:       Eklika({w_i}, S_i ∩ N(w_i));
18:       c[i] = max; // veličina maksimalne klike u S_i
19:   return;
```

Slika 5. Estergardov algoritam

Ovde je max globalna promjenljiva čija je vrednost veličina najveće trenutno pronađene klike u G. Veličina maksimalne pronađene klike u S_i registruje se u promenljivoj $c[i]$, članu niza c dužine n. Veličina $c[i]$ pronađene klike za neko i je za 1 veća od prethodne (odnosno $c[i] = c[i+1] + 1$) ako i samo ako u S_i postoji klika veličine $c[i+1] + 1$, koja sadrži čvor w_i.

Argumenti za poziv $Eklika$ (linije 1-13) su tekuća klika K i skup U čvorova za koje se zna da su povezani sa svim čvorovima iz K. U pozivu u liniji 17 je $K = \{w_i\}$ i $U = S_i \cap N(w_i)$, pa su ove pretpostavke trivijalno ispunjene. Očigledno je da je $c[n] = 1$, jer u S_n postoji klika veličine 1.

Neka je u pozivu $Eklika\ K = \{w_{i_1}, w_{i_2}, \dots, w_{i_p}\},\ \ i_1 < i_2 < \cdots < i_p,\ p \geq 1$, i neka je $U = \{w_{j_1}, w_{j_2}, \dots, w_{j_q}\},\ i_p < j_1 < j_2 < \cdots < j_q,\ q \geq 0$.

1) U slučaju kad je $q = 0$ (linije 2-4) klika Kse ne može više povećavati čvorovima iz Q, pa se provjerava da li je $p > max$, odnosno da li je K nova najveća klika, i ako jeste, to se registruje povećavanjem $max = p$.
2) U slučaju kad je $q > 0$ (linije 7-14) za $l = 1,2, \dots, p$ rekurzivno se traži proširenje klike $K' = K \cup \{w_{j_l}\}$ čvorovima iz skupa $U' = \{w_{j_{l+1}}, w_{j_{l+2}}, \dots, w_{j_q}\} \cap N(w_{j_l})$ (koji su svi povezani sa svim čvorovima iz skupa K'), pozivom $Eklika(K', U')$, sa izuzetkom u dva slučaja – kada se vrši odsijecanje, jer nema izgleda da se pronađe klika veća od max:
   A) Ako je $p + q < max$ (**odsijecanje 1:** čak i ako se svi elementi skupa Q uključe u kliku, veličina klike biće manja od max),odnosno
   B) Ako je $p + c[j_l] < max$(**odsijecanje 2:** čak i ako se klika proširi najvećom mogućom klikom u skupu S_{j_l}, veličina klike biće manja od max).

Posle i-tog prolaska kroz petlju u linijama 18-19 vrijednost max jednaka je veličini maksimalne klike u S_i koja sadrži čvor w_i, a $c[i]$ dobija vrijednost max. Po završetku te petlje veličina maksimalne klike u G je $c[1]$.

Zapaža se da se algoritam $Eklika$ razlikuje od algoritma funkcije $KPklika$ u tome što $Eklika$ ima još jednu mogućnost odsijecanja zahvaljujući postojanju niza $c[i]$. Pošto niz $c[i]$ čuva informaciju o tome kolika je maksimalna klika u skupu S_i moguće je prekinuti pretragu u momentu kada je veličina trenutne klike toliko mala da i ako je proširimo maksimalnom klikom iz skupa S_i, gdje je i najmanji indeks čvora iz skupa U, veličina klike neće biti veća od do sada pronađene maksimalne klike.

Dokaz korektnosti algoritma $Eklika$zasniva se na sljedećem tvrđenju.

Teorema 3. U toku izvršavanja algoritma $Eklika(K, U)$ svi čvorovi iz K manji su od svih čvorova iz U. Svi čvorovi iz U povezani su sa svim čvorovima iz K.

Dokaz teoreme 3 slijedi iz dokaza teoreme 2 uz narednu dopunu.

Ako u trenutku prelaska na potfamiliju ne važi da je $|K \cup U| < max$, tada ako je $c[i] + |K| < max$ $(|K_i \cup K| < max)$, pronalaženje klike veličine max u skupu$K \cup U$ je nemoguće (jer bi iz skupa U bilo dodato još maksimalno $c[i] - 1$ čvorova), pa se izvršava odsijecanje. Kako važi teorema 2, u toku izvršavanja algoritma $Eklika(K, U)$ štampaju se sve klike veličine max za svaki od skupova S_i, $i = n, n-1, \dots, 1$.

Primjer 3. Razmotrimo graf G sa skupom čvorova $\{1,2, \dots, 8\}$ prikazan na Slici 2. Primjena algoritma $Eklika$ na ovaj graf opisuje se nizom koraka u nastavku. Pri tome je dubina rekurzivnog poziva predstavljena dubinom uvlačenja odgovarajućeg reda, kao u Primjeru 1.

1: $S[8] = \{8\}$; $K = \{8\}$ $U = \{\}$; $max = 1$ Pronađena je klika $\{8\}$; $c[8] = 1$;
2: $S[7] = \{7,8\}$; $K = \{7\}$ $U = \{\}$; $max = 1$ Nova klika je $\{7\}$; $c[7] = 1$;
3: $S[6] = \{6,7,8\}$; $K = \{6\}$ $U = \{\}$; $max = 1$ Nova klika je $\{6\}$; $c[6] = 1$;
4: $S[5] = \{5,6,7,8\}$; $K = \{5\}$ $U = \{7,8\}$;
    5: $K = \{5,7\}$ $U = \{\}$; $max = 2$ Pronađena je klika $\{7,5\}$;
    6: $K = \{5,8\}$ $U = \{\}$; $max = 2$ Nova klika je $\{8,5\}$;
7: Odsijecanje $(|K \cup U| < max)$ $|K \cup U| = 1$; $c[5] = 2$;
8: $S[4] = \{4,5,6,7,8\}$; $K = \{4\}$ $U = \{6,7,8\}$;

9: $K = \{4,6\}\ U = \{\};\ max = 2$ Nova klika je {6,4};
10: $K = \{4,7\}\ U = \{\};\ max = 2$ Nova klika je {7,4};
11: $K = \{4,8\}\ U = \{\};\ max = 2$ Nova klika je {8,4};
12: Odsijecanje $(|K\ \cup\ U| < max)\ \ |K\ \cup\ U| = 1;\ c[4] = 2;$
13: $S[3] = \{3,4,5,6,7,8\};\ K = \{3\}\ U = \{6,7,8\};$
14: $K = \{3,6\}\ U = \{\};\ max = 2$ Nova klika je {6,3};
15: $K = \{3,7\}\ U = \{\};\ max = 2$ Nova klika je {7,3};
16: $K = \{3,8\}\ U = \{\};\ max = 2$ Nova klika je {8,3};
17: Odsijecanje $(|K\ \cup\ U| < max)\ \ |K\ \cup\ U| = 1;\ c[3] = 2;$
18: $S[2] = \{2,3,4,5,6,7,8\};\ K = \{2\}\ U = \{5,6,7\};$
19: $K = \{2,5\}\ U = \{7\};$
20: $K = \{2,5,7\}\ U = \{\};\ max = 3$ Pronadjena je klika {7,5,2};
21: Odsijecanje $(|K\ \cup\ U| < max)\ \ |K\ \cup\ U| = 2;$
22: Odsijecanje2 $(|K| + c[i] < max)\ \ |K| + c[6] = 2;\ c[2] = 3;$
23: $S[1] = \{1,2,3,4,5,6,7,8\};\ K = \{1\}\ U = \{3,5,6,8\};$
24: $K = \{1,3\}\ U = \{6,8\};$
25: $K = \{1,3,6\}\ U = \{\};\ max = 3$ Nova klika je {6,3,1};
26: $K = \{1,3,8\}\ U = \{\};\ max = 3$ Nova klika je {8,3,1};
27: Odsijecanje $(|K\ \cup\ U| < max)\ \ |K\ \cup\ U| = 2$
28: $K = \{1,5\}\ U = \{8\};$
29: $K = \{1,5,8\}\ U = \{\};\ max = 3$ Nova klika je {8,5,1};
30: Odsijecanje $(|K\ \cup\ U| < max)\ \ |K\ \cup\ U| = 2$
31: Odsijecanje2 $(|K| + c[i] < max)\ \ |K| + c[6] = 2;\ c[1] = 3;$

3.4. Algoritam iz programa Cliquer

Cliquer je skup C rutina za nalaženje klika u proizvoljnom težinskom grafu. Omogućuje traženje maksimalne klike, maksimalne težinske klike ili klike čija veličina ili težina je unutar datog opsega, opcionalno ograničavajući traženje maksimalne klike. Cliquer koristi tačan algoritam grananja i ograničavanja (branch-and-bound) [17] zasnovan na algoritmu Karagana i Pardalosa, za pronalaženje maksimalne klike i maksimalne težinske klike. Dodaje efikasniju metodu odsijecanja čuvajući veličinu maksimalne klike podgrafova koje je pretražio. U mnogim slučajevima, ovaj algoritam je brži od ostalih poznatih algoritama.

Algoritam iz programa Cliquer (u nastavku $Cklika$) pronalazi sve maksimalne klike u proizvoljnom neusmjernom grafu. Neka je fiksiran redoslijed čvorova $\{w_1, w_2, \dots, w_n\}$ u zadatom grafu G i neka je $S_i = \{w_1, w_2, \dots, w_i\},\ i = 1, \dots, n$. Neka je max globalna promjenljiva čija je vrijednost veličina najveće trenutno pronađene klike u G. Algoritam traži klike koje sadrže čvor w_i u skupovima S_i redom za $i = 1,2, \dots n–1, n$. Svako od tih traženja odvija se kao u algoritmu Estergarda. U programskoj realizaciji algoritam $Cklika$ se prvo iskoristi da odredi veličinu maksimalne klike, zatim pomoću modifikacije istog algoritma program nastavlja od mjesta gdje je prvi put pronašao maksimalnu kliku i redom štampa sve maksimalne klike. Algoritam $Cklika$ je prikazan na slici 7.

```
Algoritam Cklika(K, U)// svi čvorovi u skupu K su veći od svih čvorova u skupu U
1:      if U == ∅ then
2:          if |K| ≥ max then
3:              max = |K|;
4:              štampati K;
5:              nasao: =true;
6:              return
7:      while U ≠ ∅ do
8:          if |K ∪ U| < max then              // odsijecanje 1
9:              return
10:         i =najveći redni broj nekog čvora iz skupa U;
11:         if |K| + c[i] < max then           // odsijecanje 2
12:             return
13:         U = U\{w_i} ;
14:         Cklika(K ∪ {w_i}, U ∩ N(w_i))      // čvor w_i uključen je u kliku
15:         if nasao == true then
16:             return
17:     return
Program glavni;
18:     max = 0;
19:     for i = 1 to n do
20:         nasao =false;
21:         Cklika({w_i}, S_i ∩ N(w_i));
22:         c[i] = max; // veličina maksimalne klike u S_i
23:     return;
```

Slika7. Algoritam iz programa Cliquer

Veličina maksimalne pronađene klike u S_i registruje se u promenljivoj $c[i]$, članu niza c dužine n. Veličina $c[i]$ pronađene klike za neko i je za 1veća od prethodne (odnosno $c[i] = c[i-1] + 1$) ako i samo ako u S_i postoji klika veličine $c[i-1] + 1$, koja sadrži čvor w_i.

Argumenti za poziv $Cklika$ (linije 1-14) su U i K, gde je K tekuća klika, a U skup čvorova za koje se zna da su povezani sa svim čvorovima iz K. U pozivu u liniji 17 je $K = \{w_i\}$ i $U = S_i \cap N(w_i)$, pa su ove pretpostavke trivijalno ispunjene. Očigledno je da je $c[1] = 1$, jer u S_1 postoji klika veličine 1.

Neka je u pozivu $CklikaK = \{w_{i_1}, w_{i_2}, \dots, w_{i_p}\}$, $i_1 > i_2 > \cdots > i_p$, $p \geq 1$, i neka je $U = \{w_{j_1}, w_{j_2}, \dots, w_{j_q}\}$, $i_p > j_1 > j_2 > \cdots > j_q$, $q \geq 0$.

1) U slučaju kad je $q = 0$ (linije 2-4) klika K ne može se više povećavati čvorovima iz Q, pa se provjerava da li je $p > max$, odnosno da li je K nova najveća klika, i ako jeste, to se registruje povećavanjem $max = p$.
2) U slučaju kad je $q > 0$ (linije 7-13) za $l = 1,2, \dots, p$ rekurzivno se traži proširenje klike $K' = K \cup \{w_{j_l}\}$ čvorovima iz skupa $U' = \{w_{j_1}, w_{j_2}, \dots, w_{j_{l-1}}\} \cap N(w_{j_l})$ (koji su svi povezani sa svim čvorovima iz skupa K'), pozivom $Cklika(K', U')$, sa izuzetkom u dva slučaja – kada se vrši odsijecanje, jer nema izgleda da se pronađe klika veća od max:

A) Ako je $p + q < max$ (**odsijecanje 1:** čak i ako se svi elementi skupa Q uključe u kliku, veličina klike biće manja od max), odnosno
B) Ako je $p + c[j_i] < max$(**odsijecanje 2:** čak i ako se klika proširi najvećom mogućom klikom u skupu S_{j_i}, veličina klike biće manja od max).

Poslije i-tog prolaska kroz petlju u linijama 17-18 vrijednost max jednaka je veličini maksimalne klike u S_i koja sadrži čvor w_i, a $c[i]$ dobija vrijednost max. Po završetku te petlje veličina maksimalne klike u G je $c[n]$.

Zapaža se da se algoritam funkcije $Cklika$ razlikuje od algoritma funkcije $Eklika$ u tome što $Cklika$ vrši pretraživanje skupova redom $\{w_1\}$, $\{w_1, w_2\}$, $\{w_1, w_2, w_3\}$,..., $\{w_1, w_2, \dots, w_n\}$, a $Eklika$ $\{w_n\}$, $\{w_n, w_{n-1}\}$, $\{w_1, w_{n-1}, w_{n-2}\}$,..., $\{w_n, w_{n-1}, \dots, w_1\}$.

Dokaz korektnosti zasniva se na sljedećem tvrđenju.

Teorema 4. U toku izvršavanja algoritma $Cklika(K, U)$ svi čvorovi iz K veći su od svih čvorova iz U. Svi čvorovi iz U povezani su sa svim čvorovima iz K.

Dokaz se izvodi principom matematičke indukcije.

Baza indukcije: Na početku algoritma skup K je $\{w_1\}$, a skup U je prazan skup. Prema tome tvrđenje važi jer je skup U prazan.

Indukcijska hipoteza: Pretpostavimo da tvrđenje važi za poziv $Cklika(K, U)$.

Korak indukcije: Dokažimo da tvrđenje važi za poziv $Cklika\big(K \cup \{w_i\}, U \cap N(w_i)\big)$. Po pretpostavci važi da su svi čvorovi u skupu K veći od svih čvorova u skupu U. Ako je $|K \cup U| < max$, onda je pronalaženje klike veličine max u $K \cup U$ nemoguće, pa se izvršava odsijecanje. U liniji 10 bira se čvor w_i sa najvećim indeksom u skupu U. Ukoliko važi da je $c[i] + |K| < max$ ($|K_i \cup K| < max$), pronalaženje klike veličine max u skupu $K \cup U$ je takođe nemoguće (jer bi iz skupa U bilo dodato još maksimalno $c[i] - 1$ čvorova), pa se izvršava odsijecanje. U protivnom, ako nije ispunjen ni jedan od uslova koji vode ka odsijecanju, tada se čvor w_i izbacuje iz skupa U i dodaje se skupu K. Time su i dalje svi čvorovi iz skupa K veći od svih čvorova iz skupa U. Takođe svi čvorovi iz skupa $K \cup \{w_i\}$ veći od svih čvorova iz skupa $U \cap N(w_i)$. Time je dokazano da tvrđenje (i) važi za svaki poziv $Cklika\big(K \cup \{w_i\}, U \cap N(w_i)\big)$.

Niz $c[i]$ čuva vrijednost najveće klike skupa S_i. Za sve $i = 2, \dots, n$ važi da je $c[i] = c[i-1]$ ili $c[i] = c[i-1] + 1$. Takođe važi da je $c[i] = c[i-1] + 1$ ako i samo ako postoji klika u S_i veličine $c[i-1] + 1$ koja sadrži čvor w_i. Polazeći od $c[1] = 1$, traže se takve klike. Ako je klika pronađena, $c[i] = c[i-1] + 1$, inače $c[i] = c[i-1]$.
Koristeći ovaj algoritam Cliquer prvo određuje veličinu klike, a onda počinje sa pretragom ponovo od odgovarajućeg indeksa i da bi pronašao i ispisao sve maksimalne klike. Prilikom određivanja veličine maksimalne klike u uslovima odsijecanja koristi se znak $\leq$.

Primjer 4. Razmotrimo graf G sa skupom čvorova $\{1,2, \dots,8\}$ prikazan na Slici 2. Primjena algoritma $Cklika$ na ovaj graf opisuje se nizom koraka u nastavku. Pri tome je dubina rekurzivnog poziva predstavljena dubinom uvlačenja odgovarajućeg reda, kao u Primjeru 1.

1: $S[6] = \{1,2,3,4,5,6\}$; $K = \{6\}$ $U = \{1,2,3,4\}$;

2: $K = \{6,4\}\ U = \{\}$;
3: $K = \{6,3\}\ U = \{1\}$;
4: $K = \{6,3,1\}\ U = \{\};\ max = 3$ Pronađena je klika {1,3,6};
5: $|K \cup U| = 2$ Odsijecanje1;
6: $|K| + |c[2]| = 2$ Odsijecanje2; $c[6] = 3$;
7: $S[7] = \{1,2,3,4,5,6,7\};\ K = \{7\}\ U = \{2,3,4,5\}$;
8: $K = \{7,5\}\ U = \{2\}$;
9: $K = \{7,5,2\}\ U = \{\};\ max = 3$ Nova klika je {2,5,7};
10: $|K \cup U| = 2$ Odsijecanje1;
11: $K = \{7,4\}\ U = \{\}$;
12: $K = \{7,3\}\ U = \{\}$;
13: $|K \cup U| = 2$ Odsijecanje1; $c[7] = 3$;
14: S[8]={1,2,3,4,5,6,7,8}; K={8} U={1,3,4,5};
15: $K = \{8,5\}\ U = \{1\}$;
16: $K = \{8,5,1\}\ U = \{\};\ max = 3$ Nova klika je{1,5,8};
17: $|K \cup U| = 2$ Odsijecanje1;
18: $K = \{8,4\}\ U = \{\}$;
19: $K = \{8,3\}\ U = \{1\}$;
20: $K = \{8,3,1\}\ U = \{\};\ max = 3$ Nova klika je {1,3,8};
21: $|K \cup U| = 2$ Odsijecanje1;
22: $|K \cup U| = 2$ Odsijecanje1; $c[8] = 3$;

3.5. Poređenje algoritama

Algoritmi Estergarda i Cliquer su nastali od algoritma Karagana i Pardalosa pa su veoma slični. Algoritam Estergarda se razlikuje od algoritma Karagana i Pardalosa po tome što je u algritmu Estergarda uveden niz c u kome se čuvaju veličine maksimalnih klika za odgovarajuće skupove čvorova. Zahvaljujući tom nizu uveden je novi uslov $|K| + c[i] < max$ koji omogućava skraćivanje pretrage. Iz tog razloga program za algoritam Estergarda je brži od programa za algoritam Karagana i Pardalosa, za veliki broj testiranih grafova. Algoritam Cliquer je nastao od algortitma Estergarda i razlikuje se od njega po tome što najprije odredi veličinu maksimalne klike, a zatim od mjesta gdje je prvi put pronađena maksimalna klika ponavlja pretragu da bi ispisao sve maksimalne klike zadatog grafa. Takođe se razlikuje od algoritma Estergarda po tome što pretragu počinje od prvog čvora. Algoritam Cliquer je najbrži od svih algoritama koji su opisani u ovom radu.

Algoritmi Karagana i Pardalosa, Estergarda i Cliquer koriste dva skupa u pozivu funkcije koja pronalazi sve maksimalne klike i to su skupovi K i U (K je skup čvorova klike koja se trenutno formira, a U je skup čvorova koji su potencijalni kandidati da postanu članovi trenutne klike). Algoritam Brona i Kerboša koristi pored ovih skupova i skup S. Skup S obezbjeđuje da se ista klika ne ispiše dva puta, pa ovaj algoritam omogućuje ispisivanje svih neproširivih klika. Kod ostalih algoritama to se rješava na drugačiji način. Algoritmi Karagana i Pardalosa, Estergarda i Cliquer su zasnovani na poznatom algoritmu grananja i ograničavanja (branch and bound) [14] te nikada u pretrazi ne naiđu dva puta na istu kliku. Svaki skup čvorova se razmatra tačno jednom u algortimu. Algoritam Brona i Kerboša pronalazi sve neproširive klike i zbog toga odogvarajući program pokazuje mnogo lošije rezultate od ostalih programa pri testiranju grafova koji sadrže mnogo neproširljivih klika.

3.6. Heuristike za izbor redoslijeda čvorova

Termin „heuristika“ potiče od starogrčke riječi „heuriskein“ – što znači „naći“ ili „otkriti“. Heuristika je tehnika koja pokušava da nađe neka „dobra“ rješenja problema u okviru razumnog vremena, pri čemu se ne garantuje da će nađena rješenja biti optimalna, niti se može odrediti njihova bliskost optimalnom rješenju.

Redoslijed čvorova u pretrazi ima uticaj na brzinu pretrage. Heuristika bojenja čvorova može biti korišćena da se dobije dobar početni poredak čvorova.

Mnogi pokušaji poboljšanja algoritma Karagana i Pardalosa baziraju se na metodi izračunavanja gornje granice tokom pretraživanja. Takve granice se dobijaju bojenjem čvorova. U bojenju čvorova susjednim čvorovima moraju biti pridružene različite boje. Ako graf, ili indukovani podgraf, mogu biti obojeni sa, na primjer, s boja, tada graf, ili podgraf, ne može sadržati kliku veličine $s + 1$.

Heuristika korišćena u [2] za algoritam Karagana i Pardalosa sortira čvorove po stepenima čvorova tako da čvor v_1 ima najmanji stepen. Zatim vrši bojenje svih čvorova za koje je to moguće počevši od prvog u nizu sortiranih neobojenih čvorova tekućom bojom. Uzima novu boju i postupak ponavlja sve dok ima neobojenih čvorova. Pomenuta heuristika se primjenjuje na sva četiri algoritma.

Opis prve heuristike

Neka je dat graf $G = (V, E)$. Algoritam prve heuristike vrši bojenje svih čvorova datog grafa tako da svi susjedni čvorovi budu obojeni različitim bojama. Prvo se sortiraju čvorovi iz V po stepenima u opadajućem redoslijedu. Zatim se vrši bojenje svih čvorova za koje je to moguće počevši od prvog u nizu sortiranih neobojenih čvorova tekućom bojom. Provjeri se da li ima neobojenih čvorova. Ako ima, uzima se nova boja i ponavlja se isti postupak. Ako su svi čvorovi obojeni, bojenje se završava. Redoslijed biranja čvorova je redoslijed koji se koristi za traženje maksimalne klike.
Neka je N skup neobojenih čvorova sortiranih po stepenu čvorova u opadajućem redoslijedu, M pomoćni skup i Ob skup obojenih čvorova. Algoritam je opisan pseudo kodom na slici 8.

Primjer 5. Razmotrimo graf G sa skupom čvorova $\{1,2,\dots,8\}$ prikazan na Slici 2.
Matrični zapis grafa G je sljedeći:

8
0 0 1 0 1 1 0 1
0 0 0 0 1 1 1 0
1 0 0 0 0 1 1 1
0 0 0 0 0 1 1 1
1 1 0 0 0 0 1 1
1 1 1 1 0 0 0 0
0 1 1 1 1 0 0 0
1 0 1 1 1 0 0 0

Stepeni čvorova grafa redom su 4 3 4 3 4 4 4 4.

Redoslijed sortiranih čvorova je 1 3 5 6 7 8 2 4.

Redoslijed bojenja čvorova je 1 7 3 5 6 8 2 4. Čvorovi su obojeni pomoću četiri boje, što predstavlja gornju granicu veličine maksimalne klike datog grafa.

Redoslijed čvorova za traženje klike je obrnut redoslijedu bojenja čvorova 4 2 8 6 5 3 7 1.

Algoritam bojenje
1: $boja = 0$;
2: **while** $N \neq \emptyset$ **do**
3: $boja++$;
4: $i =$ indeks prvog neobojenog čvora iz skupa N;
(najmanji redni broj neobojenog čvora trenutno najvećeg stepena)
5: $N = N \backslash \{v_i\}$;
6: $Ob = Ob \cup \{v_i\}$;
7: $M = N \backslash N(v_i)$;
8: **while** $M \neq \emptyset$ **do**
9: j:= indeks prvog neobojenog čvora iz skupa M;
10: $M = M \backslash \{v_j\}$;
11: $N = N \backslash \{v_j\}$;
12: $Ob = Ob \cup \{v_j\}$;
13: $M = M \backslash N(v_j)$;

Slika 8. Algoritam prve heuristike

Opis druge heuristike

Druga heuristika se razlikuje od prve heuristike po tome što se primjenjuje na komplement polaznog grafa. Prvo se napravi komplement polaznog grafa, zatim se na njega primijeni algoritam prve heuristike, odredi se redoslijed čvorova, a zatim se taj redoslijed čvorova primijeni na polazni graf.

Heuristika može imati različite efekte za različite tipove grafova. Eksperimenti su pokazali da ova heuristika za redoslijed čvorova je efikasna za neke algoritme i neke klase grafova. Naročito se pokazala efikasnom u slučaju algoritma Brona i Kerboša.

3.7. Paralelizacija algoritama

Paralelni algoritam je algoritam koji može da se izvršava dio po dio u isto vrijeme na više različitih uređaja za obradu, zatim se ti dijelovi na kraju sastavljaju ponovo zajedno da bi se dobio tačan rezultat.Efekat paralelizacije može se mjeriti ubrzanjem algoritma. Neka je T_1 vrijeme izvršavanja programa na jednom procesoru, a T_p vrijeme izvršavanja programa na p procesora. Tada se ubrzanje definiše sa $\frac{T_1}{T_p}$.

U radu [10] prikazani su rezultati dobijeni paralelizacijom algoritma Karagana i Pardalosa. Program koristi jedan (glavni) procesor za raspoređivanje poslova ostalim (pomoćnim) procesorima. Glavni procesor raspoređuje potprobleme sporednim procesorima, koji odrađuju

zadati posao. Kada pomoćni procesor završi svoj posao, on šalje rezultat glavnom procesoru, i od njega dobija novi potproblem.

U algoritmu Karagana i Pardalosa pomoćnim procesorima se prosljeđuje čvor v_i za proširivanje tj. za traženje maksimalne klike u skupu $S_i = \{v_i, v_{i+1}, \dots, v_n\}$ koja sadrži čvor v_i. Kad pomoćni procesor završi sa datim čvorom, šalje rezultat glavnom procesoru, koji mu prosljeđuje novi čvor za obradu, ukoliko još ima čvorova za koje nisu pronađene maksimalne klike.

Algoritam je u [10] testiran na grafovima od 64 čvorova i 1824 grana, do 500 čvorova i 62130 grana. Izvršeno je upoređivanje brzine izvršavanja paralelnog algoritma sa dva i četiri procesora. Idealno ubrzanje bi bilo 3. Dobijeno je ubrzanje između 1.82 i 2.66 i poboljšava se za veće probleme. Ubrzanje za Hemingove grafove je bilo 1.27 i 2.52, a za Keller4 ubrzanje je bilo 2.41.

4. Programska realizacija algoritama

Svi programi koji se pominju u ovom poglavlju su realizovani na programskom jeziku C/C++ i nalaze se u prilogu, kao i primjeri benčmark grafova na kojima su izvršena testiranja. Funkcije koje su zajedničke za bar dva programa izdvojene su u program Pomocni. Svaki od programa u komandnoj liniji ima tri obavezna argumenta. Prvi argument je naziv datoteke iz koje se učitava polazni graf. Ta datoteka mora biti u DIMACS ascii ili DIMACS binarnom formatu (vidjeti tačku 4.7.). Drugi argument omogućuje izbor heuristike: 0-bez heuristike, 1-primjenjuje se prva heuristika, 2-primjenjuje se druga heuristika. Treći argument je znak 'p' ako treba ispisati samo prvu maksimalnu kliku ili znak 's' ako treba ispisati sve maksimalne klike.

4.1. Struktura podataka

Graf je predstavljen tipom podataka char * pokgraf što predstavlja pokazivač na matricu povezanosti u koju je upisan graf. Čvorovi grafa su numerisani sa $\{0,1,\dots,n-1\}$, gdje je n broj čvorova. Matrica povezanosti je simetrična i antirefleksivna, a i-ta vrsta matrice povezanosti nosi informaciju o tome koji su čvorovi susjedni čvoru v_i, $i=\{0,1,\dots,n-1\}$.

Skupovi K, U i S su predstavljeni pomoću nizova koji imaju broj elemenata jednak broju čvorova grafa. Ako je $K[i]=1$, tada čvor v_i pripada skupu K, $i \in \{0,1,\dots,n-1\}$. Isto važi i za ostala dva skupa.

Promjenljive koje se pominju u svim programima su

`int dim` –dimenzija grafa tj. broj čvorova grafa.

`int heuristika` -čuva informaciju o tome da li će se primjenjivati heuristika i to koja od dvije moguće

`short sveklike` -promjenljiva od koje zavisi da li će se ispisivati sve maksimalne klike(1) ili jedna(0)

`FILE *pUlaz` -pokazivač na ulaznu datoteku

`FILE *pIzlaz1` –pokazivač na izlaznu datoteku koja će biti iskorištena za upis DIMACS ascii formata grafa ili matrice povezanosti, u zavisnosti od toga da li je ulazna datoteka bila u binarnom ili ascii formatu

`FILE *pIzlaz` -pokazivač na izlaznu datoteku u kojoj će biti upisana jedna ili sve maksimalne klike

`int * stepen` -niz stepen služi za čuvanje stepena svih čvorova grafa

`int * sortirani` -u nizu sortirani čuvaju se indeksi čvorova po stepenima u opadajućem redoslijedu

`int * redoslijed` -ovaj niz čuva redoslijed bojenja čvorova

4.2. Datoteka Pomocni.cpp

U datoteci Pomocni.cpp izdvojene su sve funkcije koje koriste bar dva različita programa za pronalaženje maksimalne klike. Najprije se opisuju funkcije koje se koriste u sva četiri programa. To su funkcije koje se koriste za konverziju, funkcije za primjenu jedne od heuristika, te funkcija koja određuje broj elemenata skupa, kao i neke od funkcija za upisivanje u izlaznu datoteku izlaz.txt.

```
int konverzija(FILE * pUlaz,FILE * pIzlaz,char* pokgraf)
```

U slučaju da je ulazna datoteka bila u DIMACS ascii formatu tada se koristi ova funkcija koja čita ulaznu datoteku sa pokazivačem `FILE * pUlaz`, i u memoriji se formira matrica grafa sa pokazivačem `char* pokgraf`. Zatim se u izlaznu datoteku sa pokazivačem `FILE * pIzlaz` upisuje najprije broj čvorova grafa, a zatim i odgovarajuća matrica povezanosti.

```
void bintoascii(FILE * fp,FILE * fp1, char * name, char *name1)
```

Čita graf u DIMACS binarnom formatu iz datoteke sa imenom name i upisuje ga u novu datoteku sa imenom name1 u DIMACS ascii formatu. Ova funkcija koristi funkcije `read_graph_DIMACS_bin` i `write_graph_DIMACS_bin`

```
void read_graph_DIMACS_bin(FILE * fp,char *file)
```

Čita ulaznu datoteku u kojoj je zapisan graf u DIMACS binarnom formatu, fp je pokazivač na datoteku, a *file je pokazivač na ime datoteke koja se čita. Ova funkcija koristi funkciju `get_params`.

```
void write_graph_DIMACS_ascii(FILE * fp1, char *file)
```

Upisuje graf u datoteku sa imenom file u DIMACS ascii formatu tako što pomoću funkcije get_edge iz DIMACS binarnog formata čita grane grafa. U granama grafa uvijek je $j > i$ jer su zapisane samo grane donje trougaone matrice. fp1 je pokazivač na datoteku, a *file je pokazivač na ime datoteke u koju se upisuje.

```
int get_params()
```

Uzima broj čvorova i broj grana iz tekstualnog zaglavlja koje sadrži DIMACS ascii format. Vraća vrijednost 1 ako su broj čvorova i broj grana različiti od nule, inače vraća 0.

```
BOOL get_edge(int i, int j)
```

Ispituje da li grana (i,j) pripada grafu.

Slijede funkcije koje se koriste ako se primjenjuje jedna od heuristika.

```
int nadjiStepen(char * pokgraf,int * stepen,int dim)
```

Pronalazi stepene čvorova polaznog grafa i upisuje ih u niz stepen. Ako je `stepen[i]=m`, to znači da je stepen čvora v_i jednak m.

```
void sortirajPoStepenuCvorova(int max1, int * stepen, int * sortirani,int
dim)
```

Sortira čvorove po njihovim stepenima u opadajućem redoslijedu, a njihove indekse upisuje redom u niz sortirani. Promjenljiva max1 je maksimalni stepen čvorova grafa.

```
void komplementGrafa(char * pokgraf, int dim)
```

Ova funkcija pravi komplement polaznog grafa tako da na glavnoj dijagonali ostanu '0', a sve ostale elemente mijenja iz '1' u '0' odnosno iz '0' u '1'.

```
int najmanjiindeks(int  *niz)
```

Ova funkcija vraća indeks prvog u nizu elementa koji je različit od nula. Koristi se unutar funkcije oboji da bi se dobio indeks prvog po redu neobojenog čvora iz niza sortirani.

```
void oboji(char * pokgraf, int * sortirani,int * redoslijed,int dim)
```

Ova funkcija vrši bojenje čvorova tako što koristi niz sortirani u kome se čuvaju indeksi čvorova sortirani u opadajućem redoslijedu po stepenu čvorova. Čvorovi se boje tako da svaka dva čvora koja su povezana moraju biti obojena različitim bojama. Redoslijed bojenja čvorova se upisuje u niz redoslijed, a zatim se taj redoslijed invertuje. Da bi prilikom pretraživanja maksimalne klike mogao biti korišten ovaj redoslijed čvorova, izvrši se odgovarajuća transformacija vrsta matrice polaznog grafa.

```
int broji(int *pok, int dim)
```

Broji elemente niza sa pokazivačem * pok koji imaju vrijednost 1. Pošto se ova funkcija uvijek koristi za nizove kojima su elementi 0 ili 1, saberu se svi elementi ovog niza i taj zbir se vrati kao rezultat.

Slijede funkcije koje se koriste za upisivanje u izlaznu datoteku izlaz.txt sa pokazivačem pIzlaz.

```
void BrojLinije(int pomak,int &redni_broj,FILE * pIzlaz)
```

Upisuje redni broj linije u izlaznu datoteku izlaz.txt. kada treba da se ispišu sve maksimalne klike. Pomak je stepen uvlačenja u desno tj. predstavlja dubinu pretrage.

```
void KojaKlika(short q, int VelicinaKlike,FILE * pIzlaz)
```

Ispisuje u izlaznu datoteku o kakvoj se kliki radi i kolika je dužine te klike. Promjenljiva q pokazuje da li je pronađena klika nove maksimalne dužine ili je to nova klika iste dužine.

```
void IspisiElSkupa(int skup[],char sk,int dim,int h,int * redoslijed,FILE *
pIzlaz)
```

Ispisuje elemente skupa koji je predstavljen nizom skup. Ime skupa čiji će elementi biti ispisani se čuva u promjenljivoj sk. Promjenljliva h čuva informaciju o heuristikama. Ako se heuristika primjenjuje tada se prilikom ispisa elemenata skupa koristi niz redoslijed da bi se dobili stvarni indeksi čvorova polaznog grafa.

Sljedeću funkciju koriste svi programi osim Bron Kerboš programa.

```
void stampajKunijaU(int brojElemenataK,int brojElemenataU,FILE * pIzlaz)
```

U datoteku sa pokazivačem pIzlaz upiše koliki je broj elemenata unije skupova K i U.

Slijede funkcije koje pozivaju programi Estergard i Cliquerprogram.

```
void stampajS(int i,int dim,FILE * pIzlaz,char ch)
```

U datoteku sa pokazivačem pIzlaz upiše elemente trenutnog skupa S_i.

```
void stampajC(int i,int * pokc,FILE * pIzlaz)
```

U datoteku sa pokazivačem pIzlaz upiše vrijednost od $c[i]$ tj. veličinu maksimalne klike tenutnog skupa S_i.

```
void stampajKplusC(int brojElemenataK,int i,FILE * pIzlaz,int * pokc)
```

U datoteku sa pokazivačem pIzlaz upiše koliki je zbir broja elemenata skupa K i dužine maksimalne klike skupa S_i za razmatrani čvor i. pokc je pokazivač na niz c u kome se čuva dužina maksimalne klike skupa S_i.

Sljedeću funkciju koriste svi programi osim Cliquerprograma.

```
int minimalni(int * pok, int dim)
```

Vraća indeks prvog pronađenog člana u nizu koji ima vrijednost 1. pok je pokazivač na niz koji se pretražuje.

Sva četiri programa sastoje se od glavne funkcije main i funkcije maxklike za ispisivanje maksimalnih klika.

4.3. Algoritam Brona i Kerboša

Program BKprogram je realizacija algoritma Brona i Kerboša i nalazi se u projektu pod nazivom BronKerbos. Sastoji se od glavnog programa (glavne funkcije main) i funkcije maxklike.

```
void maxklike(int *K,int *U1,int *S1,int * redoslijed,short size,char * pokgraf)
```

Pronalazi sve maksimalne klike polaznog grafa. Pokazivači na nizove `K`, `U1` i `S1` predstavljaju skupove `K`,`U` i `S` koji su argumenti funkcije $BKklika$ algoritma Brona i Kerboša. `size` čuva veličinu trenutne klike koja se traži, a `pokgraf` je pokazivač na matricu u kojoj je zapisan polazni graf. Niz `redoslijed` se koristi samo u slučaju jedne od heuristika. Klike se ispisuju redom kojim se pronalaze sve dok se ne pronađe maksimalna klika, zatim se ispisuju sve maksimalne klike, ako ih sve treba ispisati. Da bi program ispisivao samo klike maksimalne veličine prvo treba odrediti veličinu maksimalne klike.

4.4. Algoritam Karagana i Pardalosa

Program KPprogram je realizacija algoritma Karagana i Pardalosa i nalazi se u projektu pod nazivom KaraganPardalos. Takođe se sastoji od glavne funkcije main i funkcije maxklike.

```
void maxklike(int U[], short size, char * pokgraf,int * redoslijed)
```

Niz `U` predstavlja skup čvorova koji se mogu dodati u skup `K` tj. mogu postati elementi trenutne klike koja se traži. `size` čuva veličinu trenutne klike koja se traži, a `pokgraf` je pokazivač na matricu u kojoj je zapisan polazni graf. Niz `redoslijed` se koristi samo u slučaju jedne od heuristika zbog ispisa elemenata klike. Klike se ispisuju redom kojim se pronalaze sve dok se ne pronađe maksimalna klika, zatim se ispisuju sve maksimalne klike, ako ih sve treba ispisati. Da bi program ispisivao samo klike maksimalne veličine prvo treba odrediti veličinu maksimalne klike.

4.5. Algoritam Estergarda

Program Eprogram je realizacija algoritma Estergarda i nalazi se u projektu pod nazivom Estergard. Takođe se sastoji od glavne funkcije main i funkcije maxklike.

```
void maxklike(int U[], short size, char * pokgraf,int * redoslijed,int *pokc)
```

Ispisuje sve maksimalne klike. Argumenti ove funkcije imaju isto značenje kao i kod funkcije `maxklike` u KPprogramu. Ovdje se još pojavljuje argument `pokc` koji predstavlja pokazivač na niz `c` koji čuva dužine maksimalnih klika za sve skupove S_i , $i \in \{0,1,\dots,n-1\}$.

4.6. Algoritam iz programa Cliquer

Program Cliquer predstavlja programsku realizaciju algoritma iz programa Cliquer. Sastoji se iz glavne funkcije main i funkcija izdvojiNajveci, dmaxklike i maxklike.

```
int izdvojiNajveci(int U[m])
```

Vraća indeks posljednjeg člana u nizu koji ima vrijednost 1. Koristi se u funkcijama dmaxklike i maxklike da bi se odredio čvor sa maksimalnim indeksom u skupu U . Taj čvor će biti dodat u trenutnu kliku.

```
void dmaxklike(int U[m],short size, short nadji,int pocni, char * pokgraf)
```

Određuje dužinu maksimalne klike. `U`, `size` i `pokgraf` imaju isto značenje kao i u prethodnom programu. Promjenljiva `nadji` prikazuje da li je nova maksimalna klika nađena (TRUE=1) ili nije (FALSE=0). Promjenljiva `pocni` je indeks skupa Si u kojem je prvi put pronađena maksimalna klika. Program Cliquer prvo odredi dužinu maksimalne klike pomoću funkcije dmaxklike, a zatim ako treba ispisati samo jednu kliku, jednim prolazom kroz skup S_i u kojem je prvi put pronađena maksimalna klika pronalazi maksimalnu kliku i ispisuje je.

4.7. Programi za konverziju

Benčmark grafovi

DIMACS fajl format je uobičajeni format za opisivanje benčmark grafova. Grafovi mogu takođe biti u čitljivom ASCII formatu [4,18] ili u binarnom formatu [4,16]. Binarni oblik zauzima manje prostora za grafove sa gustinom većom od 1.2%.

ASCII format

ASCII fajl se sastoji od linija teksta sa poljima koja su razdvojena najmanje jednom prazninom. Prvo polje svake linije sadrži jedan karakter i opisuje tip linije. Postoje tri vrste linija teksta:

c Comment line

Linije koje počinju sa ‘c’ su komentari i one se ignorišu.

p FORMAT NODES EDGES

Linija koja počinje sa ‘p’ opisuje dimenziju grafa. Polje FORMAT sadrži riječ “edge”. U polju NODES je zapisan broj čvorova grafa, a u polju EDGES broj linija grafa.

e W V

Linija koja počinje sa ‘*e*’ zadaje granu (W, V) $(W > V)$.

Binarni format

Fajl u binarnom formatu sastoji se od tri dijela: prve linije, tekstualnog zaglavlja i binarnog bloka. U prvoj liniji je zapisana dužina tekstualnog zaglavlja, u karakterima. Zaglavlje sadrži iste linije kao i ASCII format, osim linija koje počinju sa ‘e’. Binarni blok sadrži donji trougaoni dio matrice povezanosti datog grafa u binarnom formatu. Pomoću funkcije `bintoascii` benčmark grafovi se iz binarnog prevode u ASCII format.

Grafovi

Graf može biti predstavljen i u formatu simetrične matrice povezanosti. Ovaj fajl se sastoji od dva dijela. U prvoj liniji je zapisan broj čvorova grafa. U linijama koje slijede zapisuju se

redom vrste simetrične matrice povezanosti grafa. Elementi na glavnoj dijagonali su jednaki nuli, a za ostale elemente važi da je $a_{i,j} = 1$, ako $(i, j) \in E$, inače $a_{i,j} = 0$.

Bin2asc[15] je program koji prevodi benčmark grafove date u binarnom zapisu u ASCII zapis. Program uz manje izmjene dobro radi i na Windows platformi. Izmijenjeni program koji se koristi kao funkcija u ovom radu se zove `bintoascii`. Funkcija `konverzija` prevodi graf zapisan u ASCII formatu u matricu povezanosti. Programi za pronalaženje maksimalniih klika koriste grafove zapisane u formatu simetrične matrice povezanosti.

5. Eksperimentalni rezultati

Algoritmi za pronalaženje maksimalne klike u grafu koji su opisani u poglavlju 2 su testirani na benčmark grafovima koji su opisani u 5.1.

5.1. Opis benčmark grafova

Kao testni primjeri korišćene su sljedeće familije grafova:

- **Brock** grafovi [14,18] nastaju „sakrivanjem“ klike u grafu, pri čemu je veličina klike mnogo manja od očekivane. Brokington (Brockington) i Kalberson (Culberson) su autori generatora za formiranje ovih grafova .
- **CFat** su grafovi [14,16] za problem klike nastali iz teorije o problemu otkrivanja greške (fault diagnosis problems). Za dati parameter c, ″ krug debljine c ″ (c_fat ring) je graf $G = (V, E)$ definisan na sljedeći način:

  Neka je

$$k = \left\lfloor \frac{|V|}{c \log|V|} \right\rfloor$$

  i neka su $W_0, W_1, \dots, W_{k-1}$ particije V takve da važi

$$c \log|V| \leq |W_i| \leq 1 + \lceil c \log|V| \rceil, i = 0,1, \dots, k-1$$

  Za $u \in W_i$ i $v \in W_j$ važi da $(u, v) \in E$ ako i samo ako $u \neq v$ i $|i - j| \in \{0,1, k-1\}$ [1].

- **DSJC** grafovi [17] su standardni (n, p) slučajni grafovi, gdje je n-broj čvorova, p-vjerovatnoća da dva čvora u grafu budu povezana. Korišćeni su u radu [5].
- **Frb** grafovi [20] imaju n disjunktnih klika, od kojih svaka klika ima n^α čvorova ($\alpha > 0$ je konstanta). Slučajno se biraju dvije različite klike i zatim se generiše bez ponavljanja $pn^{2\alpha}$ slučajnih grana između tih dviju klika (p je konstanta, $0 < p < 1$). Prethodni korak se ponavlja za ostale parove klika $rn \ln n - 1$ puta ($r > 0$ je konstanta).
- **Hamming** je familija grafova [14,16] zasnovana na problemu u vezi sa kodovima za ispravljanje grešaka [4]. Hemingova udaljenost između binarnih vektora $u = (u_1, u_2, \dots, u_n)$ i $v = (v_1, v_2, \dots, v_n)$ je broj indeksa i takvog da je $1 \leq i \leq n$ i $u_i \neq v_i$. Hemingova udaljenost se označava sa $dist(u, v)$. Maksimalan broj binarnih vektora veličine n koji imaju hemingovu udaljenost d označava se sa $A(n, d)$. Hemingov graf [14,16] $H(n, d)$, sa parametrima n i d, definiše se kao graf sa skupom čvorova binarnih vektora dužine n, u kojem su dva čvora susjedna ako je njihova Hemingova udaljenost najmanje d. Veličina maksimalne klike u $H(n, d)$ je $A(n, d)$. Graf $H(n, d)$ ima 2^n čvorova, $2^{n-1} \sum_{i=d}^{n} \binom{n}{i}$ grana i stepen svakog čvora je $\sum_{i=d}^{n} \binom{n}{i}$. Može se primjetiti da grafovi $H(n, n-2)$ imaju maksimalnu kliku veličine 2^{n-1}. Autor ovih grafova je Pardalos.
- **Johnson** grafovi [14,16] su takođe bazirani na problemu u vezi sa kodovima za ispravljanje grešaka. Johnson graf $J(n, w, d)$ sa parametrima n, w, d je graf sa skupom čvorova binarnih vektora dužine n i težine w, gdje su dva čvora susjedna ako je njihova nahemingova udaljenost najmanje d.Maksimalan broj binarnih vektora veličine n koji imaju tačno w jedinica i takvih da je hemingova udaljenost bilo koja dva od ovih vektora jednaka

d se označavasa $A(n,w,d)$. Veličina maksimalne klike u grafu $J(n,w,d)$ je jednaka $A(n,w,d)$.Graf $J(n,w,d)$ima $\binom{n}{w}$ čvorova, $\frac{1}{2}\binom{n}{w}\sum_{k=\left\lceil\frac{d}{2}\right\rceil}^{w}\binom{w}{k}\binom{n-w}{k}$grana, a stepen svakog čvora je $\sum_{k=\left\lceil\frac{d}{2}\right\rceil}^{w}\binom{w}{k}\binom{n-w}{k}$. Autor ovih grafova je Pardalos [4].

- **Keller** grafovi [14,16] su bazirani na Kelerovoj hipotezi o pokrivanju hiperkockama Euklidskog prostora R^n [4]. Kelerov graf I^n je graf sa skupom čvorova $V_n = \{(d_1, d_2, \dots, d_n): d_i \in \{0,1,2,3\}, i = 1,2,\dots,n\}$ gdje su dva čvora $u = (d_1, d_2, \dots, d_n)$ i $v = (d_1', d_2', \dots, d_n')$ u V_n susjedna ako i samo ako važi da

$\exists i, 1 \le i \le n: d_i - d_i' \equiv 2 \bmod 4$ i $\exists\, j \ne i, 1 \le j \le n: d_j \ne d_j'$. Kelerov graf ima 4^n čvorova, $\frac{1}{2}4^n(4^n - 3^n - n)$grana i stepen svakog čvora je $(4^n - 3^n - n)$. Može se dokazati da je veličina maksimalne klike u I^n manja ili jednaka 2^n. Autor ovih grafova je Šor (Shor) [7].

- **MANN** grafovi [14,16] dobijaju se preformulisanjem Štajnerovog problema trojke u problem klike. Autor ovih grafova je Manino(Mannino).
- **Phat** su slučajni grafovi [14,16] generisani sa $phat$ generatorom koji predstavlja generalizaciju klasičnog ravnomjernog generatora slučajnih grafova. Koriste tri parametra *n* broj čvorova, i a i b dva parametra koji se odnose na gustinu i za koje mora da važi $0 \le a \le b \le 1$. Generiše instance problema koji imaju veći raspon stepena čvorova i veće klike. Autori ovih grafova su Soriano(Soriano) i Gendrou(Gendreau) [12].
- **San** primjeri [14,16] su bazirani na radu Sančiz (Sanchis) [11]. Generator generiše primjere sa poznatom veličinom klike.
- **SanR** grafovi [14,16] su slučajni primjeri slične veličine kao u San.

5.2. Rezultati

Programi su testirani na računaru sa procesorom Core 2 Duo T5670 (2M Cache, 1.80 GHz, 800 MHz FSB), i memorijom RAM 2048MB DDR2 667MHz.

Dobijeni rezultati prikazani su u tabeli 1. Svaka vrsta u tabeli sadrži rezultate za jedan od benčmark grafova opisanih u tački 5.1. Za svaki graf najprije se navode osnovne karakteristike: broj čvorova, broj grana, gustina, veličina maksimalne klike i broj maksimalnih klika. Svaki graf obrađen je 12 puta – sa četiri programa (BKprogram, KPprogram, Eprogram, Clprogram), a za svaki program na tri načina (bez primjene heuristike, sa prvom, odnosno drugom heuristikom). U odgovarajućem polju navodi se vrijeme za koje su pronađene sve maksimalne klike. Ukoliko je polje prazno, to znači da je program prekinut, jer nije završio sa radom za jedan sat. U tabeli je svijetlo plavom bojom za svaki algoritam označen najbolji rezultat dobijen nekom od heuristika (ili bez heuristike). Ljubičastom bojom je označen najbolji rezultat u vrsti, odnosno za konkretan benčmark graf. Slovo H u tabeli označava heuristiku. U posljednjoj koloni tabele je upisano vrijeme za koje program Cliquer pronađe sve maksimalne klike za odogovarajući benčmark graf.

Najviše najboljih rezultata ima algoritam iz programa Cliquer, tj. odgovarajući program Clprogram je u većini slučajeva najbrže pronalazio maksimalne klike. Eprogram ima rezultate bliske najboljim, pa čak u nekoliko primjera i najbolje. Kada posmatramo rezultate

KPprograma, oni nisu tako dobri kao rezultati Eprograma i Clprograma, ali su mnogo bolji od BKprograma, što je razumljivo, jer BKprogram traži sve neproširive klike. Eprogram, Clprogram, KPprogram ne zaostaju mnogo u rezultatima od programa Cliquer na primjeru familija benčmark grafova c_fat, dsjc, kao i na primjeru hamming grafova sa malim brojem čvorova.

Rezultati su pokazali da redoslijed čvorova utiče na brzinu izvršavanja algoritma, te su programi testirani za dvije heuristike.

Prva heuristika izvrši sortiranje čvorova po njihovim stepenima u opadajućem redoslijedu, zatim vrši bojenje svih čvorova za koje je to moguće počevši od prvog u nizu sortiranih neobojenih čvorova tekućom bojom, uz uslov da dva susjedna čvora ne mogu biti obojena istom bojom. Za redoslijed čvorova grafa se uzima inverzan redoslijed u odnosu na onaj koji je dobijen bojenjem grafa. Ova heuristika se pokazala efikasnom u slučaju Eprograma i Clprograma, programa koji su bili najbolji i bez heuristike, naročito kod grafova sa malim brojem čvorova. KPprogram je takođe imao bolje rezultate kada je korišćena prva heuristika za veliki broj testiranih benčmark grafova.

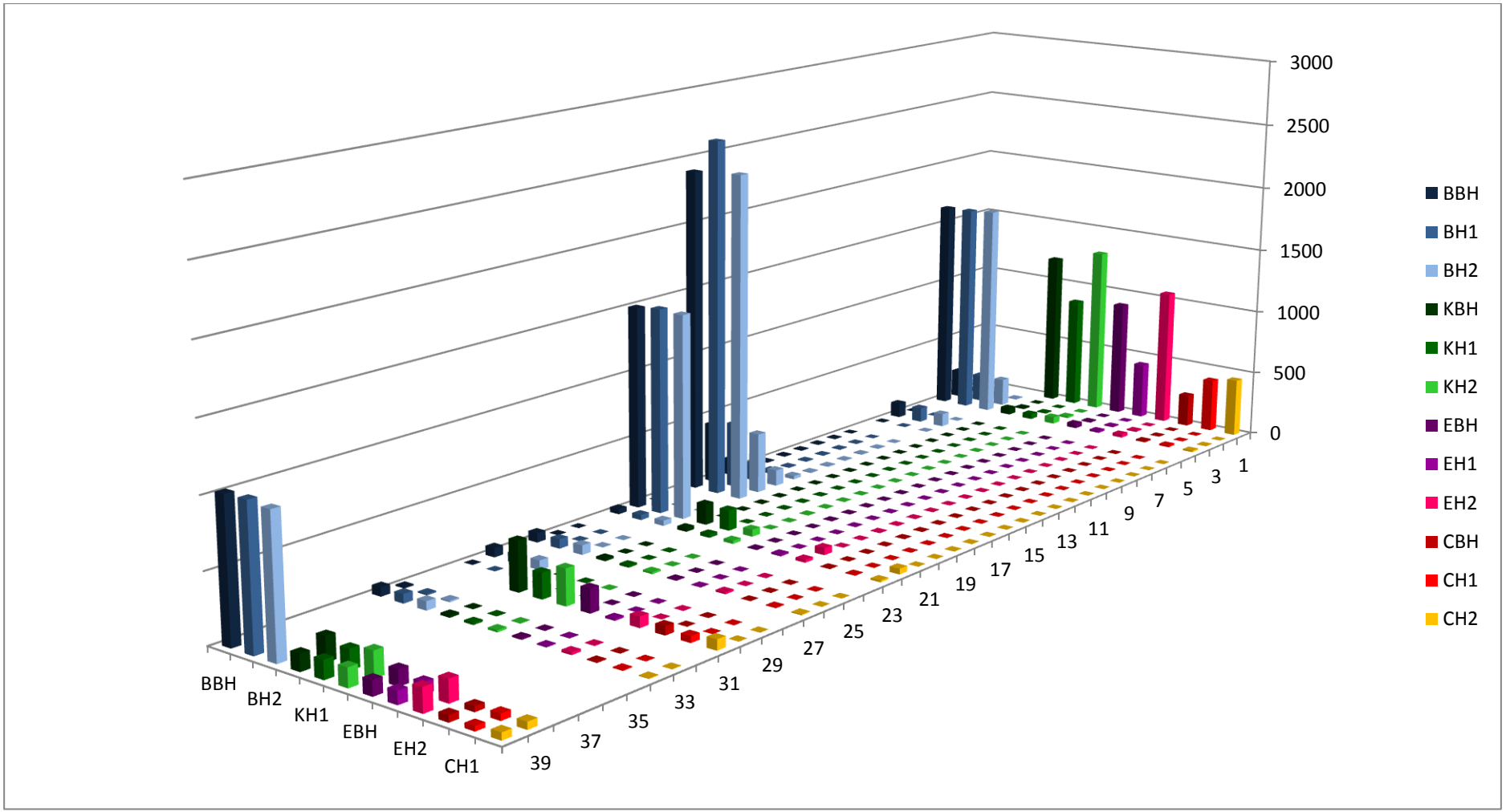
Druga heuristika se razlikuje od prve po tome što isti postupak primjeni na komplement polaznog grafa. Zanimljivo je da ova heuristika dobro radi za BKprogram, koji je imao najlošije rezultate.

Ako želimo koristiti jednostavan program za traženje svih maksimalnih klika rezultati sugerišu da je najbolje koristiti Clprogram sa prvom heuristikom.

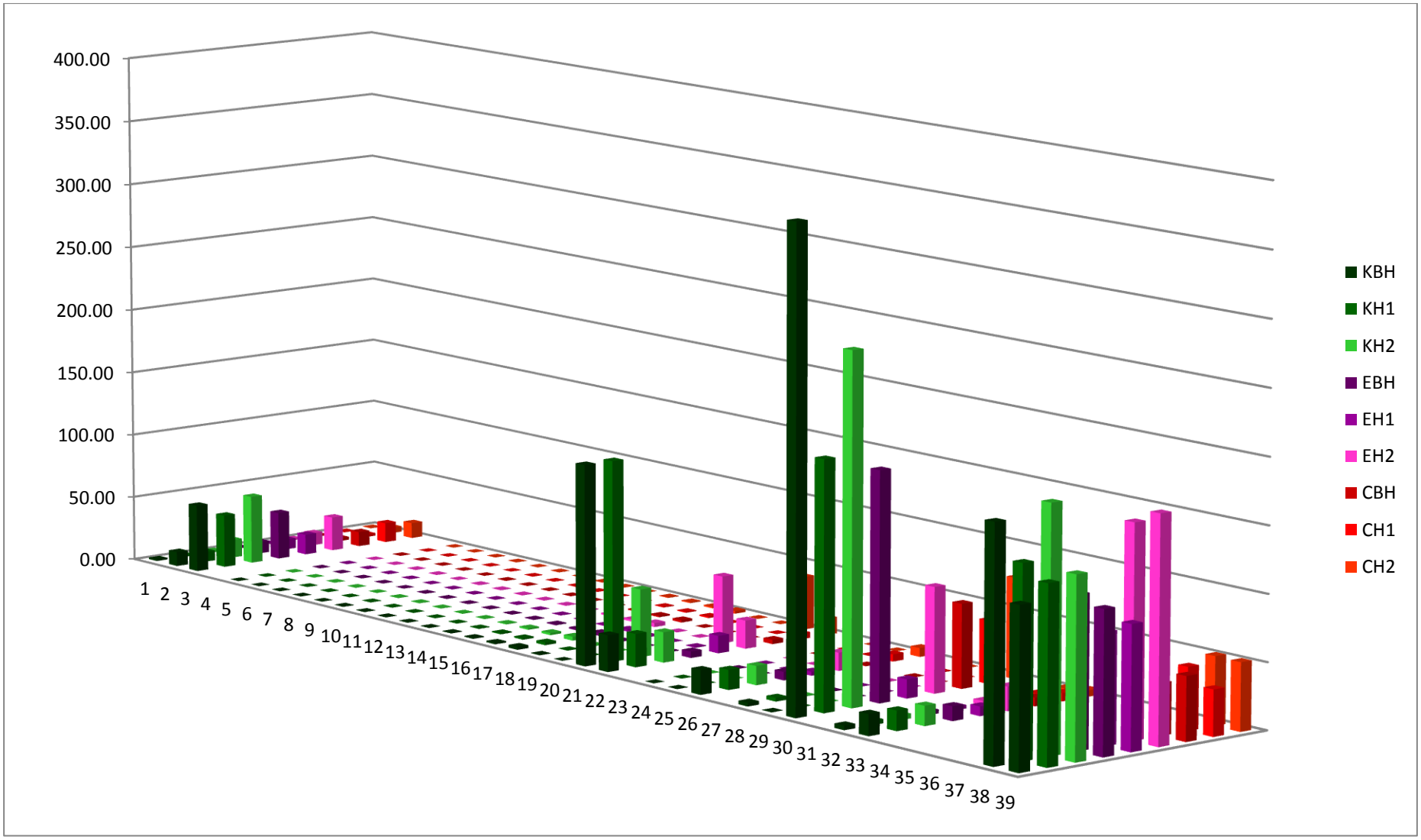
Graf	V (broj čvorova)	E (broj grana)	Gustina	klike	Broj maksimalnih klika	BKprogram			KPprogram			Eprogram			Clproram			Cliquer
						Bez H	H1	H2	Bez H	H1	H2	Bez H	H1	H2	Bez H	H1	H2	
brock200_1	200	14834	0.75	21	2				1192.33	861.95	1300.28	898.20	432.61	1058.54	248.85	409.23	447.67	39.28
brock200_2	200	9876	0.50	12	1	7.44	7.57	7.41	1.31	0.94	1.30	0.64	0.76	0.97	0.36	0.32	0.33	0.02
brock200_3	200	12048	0.61	15	1	212.60	211.16	211.98	11.13	7.92	14.09	9.19	7.85	9.79	2.37	2.13	4.41	0.24
brock200_4	200	13089	0.66	17	1	1642.09	1647.38	1664.09	52.22	41.02	52.80	36.57	15.79	26.44	11.33	15.22	12.05	0.88
brock400_2	400	59786	0.75	29	1													9987.32
cfat200-1	200	1534	0.08	12	14	0.14	0.11	0.11	0.04	0.04	0.04	0.02	0.02	0.03	0.07	0.03	0.04	0.00
cfat200-2	200	3235	0.16	24	1	113.50	111.69	100.78	0.02	0.02	0.05	0.03	0.03	0.05	0.02	0.02	0.03	0.00
cfat500-1	500	4459	0.04	14	19	1.55	1.46	1.36	0.10	0.10	0.13	0.09	0.10	0.13	0.09	0.09	0.13	0.00
cfat500-2	500	9139	0.07	26	19				0.10	0.11	0.17	0.13	0.13	0.18	0.11	0.10	0.16	0.00
dsjc125	125	736	0.09	4	19	0.06	0.02	0.02	0.06	0.02	0.03	0.01	0.01	0.01	0.01	0.01	0.02	0.00
frb30-1	450	17827	0.18	15	30	5.07	5.24	5.03	0.28	0.23	0.26	0.22	0.14	0.25	0.21	0.15	0.19	0.04
frb30-2	450	17874	0.18	15	33	5.63	6.00	5.73	0.21	0.23	0.26	0.22	0.16	0.25	0.18	0.16	0.20	0.03
frb30-3	450	17809	0.18	15	32	5.28	5.49	5.21	0.21	0.22	0.28	0.20	0.14	0.26	0.18	0.12	0.18	0.03
frb30-4	450	17831	0.18	16	4	6.18	6.64	6.37	0.20	0.20	0.26	0.21	0.16	0.25	0.16	0.12	0.18	0.02
frb30-5	450	17794	0.18	15	34	5.87	6.21	5.82	0.23	0.23	0.26	0.21	0.15	0.30	0.20	0.12	0.20	0.02
frb35-1	595	27856	0.16	17	35	25.62	27.39	25.01	0.42	0.43	0.53	0.40	0.26	0.51	0.37	0.22	0.46	0.04
frb40-1	760	41314	0.14	19	41	120.80	128.29	118.73	0.71	0.86	1.13	0.70	0.51	1.14	1.01	0.51	0.73	0.08
frb45-1	945	59186	0.13	21	45	442.54	485.89	447.13	1.17	1.16	1.67	1.19	0.67	1.54	1.14	0.63	1.19	0.11
frb50-1	1150	80072	0.12	23	50	2405.79	2657.80	2436.57	2.24	2.07	3.01	2.34	1.43	2.98	1.84	1.06	2.03	0.17
hamming6-2	64	1824	0.90	32	2				0.37	0.30	0.11	0.01	0.01	0.11	0.02	0.01	0.07	0.00

hamming6-4	64	704	0.35	4	240	0.02	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.02	0.01	0.04	0.00
hamming8-4	256	20864	0.64	16	480	1502.46	1525.75	1518.28	152.51	152.29	51.57	0.60	0.60	51.28	0.05	0.05	40.68	0.00
johnson16-2-4	120	5460	0.76	8	2027025	38.95	37.97	39.19	27.82	25.36	23.16	4.92	13.36	21.59	3.31	4.53	12.75	11.30
johnson32-2-4	496	107880	0.88	16														
johnson8-2-4	28	210	0.56	4	105	0.06	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
johnson8-4-4	70	1855	0.77	14	30	2.35	2.27	2.34	0.13	0.14	0.06	0.02	0.02	0.06	0.01	0.01	0.03	0.00
keller4	171	9435	0.65	11	2304	68.84	68.78	70.30	18.61	15.19	14.50	6.47	3.00	13.84	0.54	5.50	6.76	0.69
MANN_a27	378	70551	0.99	126														
MANN_a9	45	918	0.93	16	9540	68.02	68.16	69.92	2.76	2.82	0.68	0.48	0.51	0.60	0.22	0.53	0.27	0.09
p_hat300-1	300	10933	0.24	8	13	0.63	0.64	0.65	0.30	0.30	0.25	0.22	0.20	0.28	0.13	0.11	0.14	0.01
p_hat300-2	300	21928	0.49	25	52				363.93	187.56	264.45	173.12	14.41	79.83	63.79	47.64	76.23	1.63
p_hat300-3	300	33390	0.74	36	10													2055.95
p_hat500-1	500	31569	0.25	9	78	10.70	10.65	10.69	3.40	3.20	3.56	2.95	2.65	3.29	1.39	1.54	1.79	0.23
p_hat700-1	700	60999	0.25	11	2	65.84	66.86	65.52	16.35	15.21	15.02	10.98	7.18	18.31	6.41	6.17	4.77	0.32
san200_0.7_1	200	13930	0.70	30	1													2.31
san200_0.7_2	200	13930	0.70	18	2													0.00
san200_0.9_1	200	17910	0.90	70	1													0.19
san200_0.9_2	200	17910	0.90	60	1													26.05
sanr200_0.7	200	13868	0.70	18	13				176.05	143.27	183.23	113.82	83.13	159.26	38.65	47.06	51.40	11.63
sanr400_0.5	400	39984	0.50	13	4	1020.69	1026.06	1008.75	120.71	132.76	135.77	107.39	93.08	169.65	48.24	34.81	51.18	4.48

Tabela1. Rezultati testiranja programa za pronalaženje maksimalnih klika u grafu.



Slika 9. Grafički prikaz tabele1.



Slika 10. Grafički prikaz tabele1, bez rezultata BKprograma

6. Zaključak

U radu su opisana i programski realizovana četiri algoritma za pronalaženje maksimalnih klika (algoritam Brona i Kerboša, Karagana i Pardalosa, Estergarda i algoritam iz programa Kliker). Izvršeno je poređenje pomenutih programa na određenim benčmark grafovima koji su opisani u tački 5.1.

Poređenjem rezultata rada četiri pomenuta programa na primjerima benčmark grafova može se vidjeti da je najbrži program za algoritam iz programa Cliquer. Za njim ne zaostaje mnogo program za algoritam Estergarda. Nešto slabije rezultate je pokazao program za algoritam Karagana i Pardalosa, a najlošije program za algoritam Brona i Kerboša koji je i najstariji od pomenutih algoritama. Program za algoritam Brona i Kerboša pronalazi sve neproširive klike u grafu i nema mogućnost odsijecanja kao ostali algoritmi, te je zbog toga pokazao najlošije rezultate. Svaki od algoritama je testiran bez promjene ulaza, kao i sa dvije heuristike za izbor redoslijeda čvorova koje se baziraju na pohlepnom algoritmu za bojenje čvorova. Za redoslijed čvorova grafa se uzima inverzan redoslijed u odnosu na onaj koji je dobijen bojenjem grafa. Druga heuristika se razlikuje od prve po tome što se bojenje grafa primjenjuje na komplement testiranog grafa. Kod programa koji su imali bolje rezultate (KPprogram, Eprogram i Clprogram) efikasna se pokazala prva heuristika, dok je za BKprogram efikasna bila druga heuristika.

Ako želimo koristiti jednostavan program za traženje svih maksimalnih klika rezultati su pokazali da je od pomenuta četiri algoritma najbolje koristiti Clprogram sa prvom heuristikom.

Program KPprogram bi se mogao paralelizovati, jer kod njega nema zavisnosti preko c[i].

Literatura

[1] Berman, Pelc, *Distributed Fault Diagnosis for Multiprocessor Systems*, Proceedings of the 20th Annual Symposium on Fault-Tolerant Computing, Vol. (1990) 340-346.

[2] R. Carraghan, P.M. Pardalos, *An exact algorithm for the maximum clique problem*, Oper. Res. Lett. **9** (1990) 375–382.

[3] F. Cazals, C. Karande, *A note on the problem of reporting maximal cliques*,Theoretical Computer Science **407** (2008), 564-568

[4] D.-Z. Du, P. M. Pardalos, *Handbook of combinatorial optimization*, Kluwer Academic Publishers, (1999), 2-47.

[5] David Johnson, Aragon, McGeoch, Schevon, *Optimization by Simulated Annelaing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning*, Operations Research, 31, 378-406 (1991).

[6] Ina Koch, *Enumerating all connected maximal common subgraphs in two graphs*, Theoretical Computer Science **250** (2001), 1-30

[7] J.C. Lagarias, P.W. Shor, *Keller's Cube-Tiling Conjecture is False in High Dimensions*, Bulletin of the AMS, 27: Vol.(1992) 279-283.

[8] S. Niskanen, Patric R.J. Östergärd, *Cliquer User's Guide*, 2003

[9] Patric R. J. Östergärd, *A fast algorithm for the maximum clique problem*, Discrete Applied Mathematics, Vol. 120(2002), 197-207.

[10] Panos M.Pardalos, Jonas Rappe, Mauricio G.C. Resende, *An exact parallel algorithm for the maximum clique problem*, High performance algorithms and software in nonlinear optimization Kluwer Academic Publishers, Vol (1998), 1-17.

[11] Laura Sanchis ,*Test Case Construction for Vertex Cover Problem*, DIMACS Workshop on Computational Support for Discrete Mathematics (1992)

[12] P. Soriano, M. Gendreau, *Solving the Maximum Clique Problem Using a Tabu Search Approach*, Annals of Operations Research 41, Vol. (1993) 385-403.

[13] Miodrag Živković, Algoritmi, Vol()

[14] http://www.cs.hbg.psu.edu/benchmarks/clique.html

[15] ftp://dimacs.rutgers.edu/pub/challenge/graph/translators/binformat/ANSI/binformat.shar

[16] ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique/

[17] http://mat.gsia.cmu.edu/COLOR/instances.html

[18] http://www.cs.sunysb.edu/~algorith/implement/dimacs/distrib/color/graph/test/

[19] http://www.mutah.edu.jo/userhomepages/CS252/branchandbound.html

[20] http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm