Minimum Edge Coloring

Marija Vučetić

Ivana Ivaneža

Uvod

Definicija problema

Neka je graf G(V,E) prost neorijentisani graf, gde je V broj čvorova i E broj ivica. Prost graf podrazumeva da ne postoje višestruke ivice koje spajaju ista dva čvora kao ni da ne postoje petlje. Grane se boje tako da nikoje dve grane koje se susreću u istom čvoru nisu obojene istom bojom.

Stepen čvora je broj grana koje su vezane za taj čvor. Maksimalni stepen grafa u oznaci Δ uzima vrednost maksimalnog stepena čvora tog grafa. Minimalan broj boja kojom ivice grafa mogu da se oboje, poznat i kao hromatski indeks $\chi'(G)$, je ili $\chi'(G) = \Delta$ ili $\chi'(G) = \Delta + 1$.

Tada graf klasifikujemo kao klasa I ili klasa II, redom. Odlučivanje da li je graf klase I ili II je NP težak problem.

Istorijat

Početak problema hromatskog indeksa počinje 1880. godine kada Tait[1] uvidja da problem bojenja ravinske mape može da se svede na bojenje ivca sa tri boje gde dve ivice koje se susreću u čvoru ne smeju biti obojene istom bojom. Sve do 1949. ovaj problem pada u zaborav dok Shannon[2] nije pokazao da ako je stepen grafa Δ, najmanji broj boja potrebnih za bojenje ivica grafa ne prelazi $3/2\ \Delta$. Veliko otkriće je imao Ruski matematičar Vadim

[1] Peter Guthrie Tait (28 April 1831 – 4 July 1901)
[2] Claude Elwood Shannon (April 30, 1916 – February 24, 2001)

G. Vizing[3] 1964. godine gde je dokazao da broj boja ivica mora biti Δ ili Δ+1. Ovde dolazimo do nastanka dve klase, u jednu spadaju grafovi čije ivice mogu da se oboje sa Δ boja, u drugu grafovi sa Δ + 1 boja. Holyer[4] je 1981. dokazao da za graf čiji je stepen Δ = 3, nalaženje da li je $\chi'(G) = 3$ ili $\chi'(G) = 4$ je NP kompletan problem. Takodje je dao pretpostavku da je problem kompletan i za Δ = k, a Daniel Leven i Zvi Galil su 1983. dokazali da jeste.

Neke od propratnih definicija za razne grafove:

1. Svi bipartitivni grafovi su u klasi 1 (dokazao Kőnig[5])
2. Kn(kompletan graf) je klase 1 ili 2 u zavisnosti da li je n paran ili neparan, redom.
3. Ako je graf regularan neparnog reda, tada je graf druge klase.
4. Za neke grafove poput bipartitivnih i visoko stepenih planarnih grafova hromatski indeks je uvek Δ.

[3] Vadim Georgievich Vizing (25 March 1937 – 23 August 2017)
[4] Ian J. Holyer - The NP-completeness of edge coloring, SIAM J. Cornput. 10 (1981),
[5] Dénes Kőnig (September 21, 1884 – October 19, 1944)

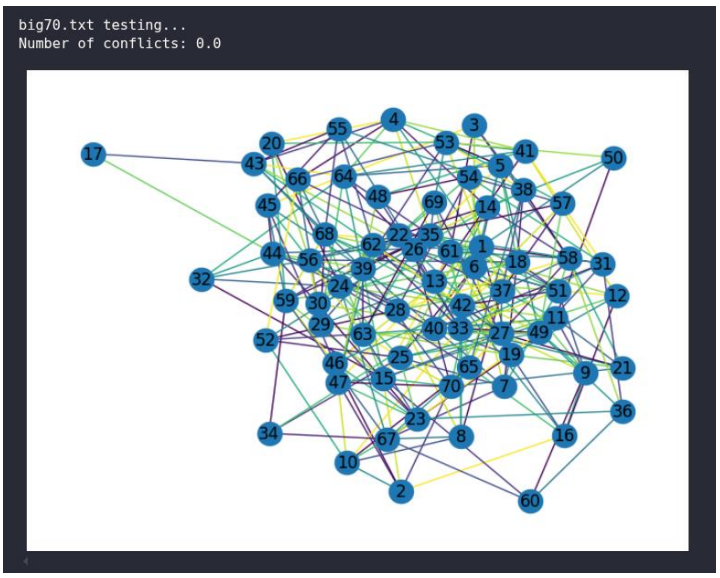
ILP rešavač

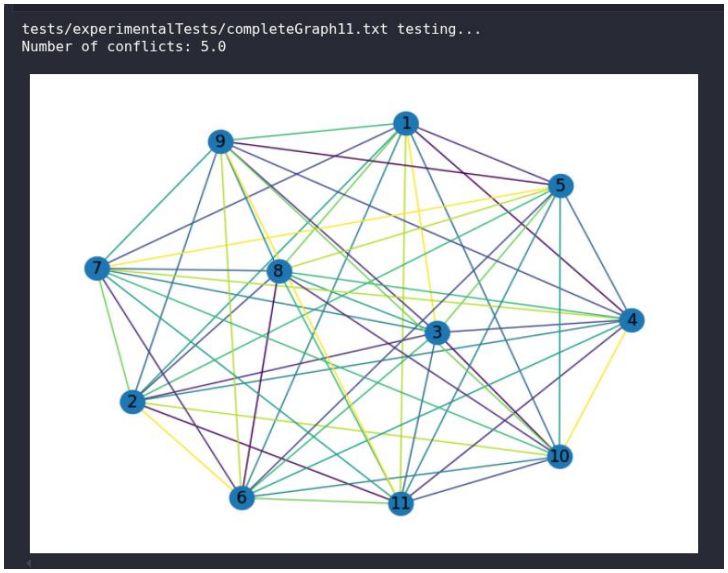
Korišćen je CPLEX optimisation studio 22.1.1.

U kodu su korišćene promenljive x[edge, colour] kao i pomoćne promenljive y[node, colour] koje predstavljaju koliko dodatnih grana ima istu boju na čvoru.
Ograničenja:

1) Svaka grana mora imati tačno jednu boju.
2) Brojanje konflikata, broj grana boje c na čvoru v:
   ako je samo jedna grana boje c, y=0 . Inače, broj konflikata = suma incidentnih ivica – 1

Cilj je minimizovati broj konflikata.
Ovakav pristup ekzaktnog rešavanja problema se ispostavio kao izuzetno jaka solucija. Čak je uspeo rešiti i guste grafove sa velikim brojem ivica, te dao zadatak optimizacionim metodama.





Optimizacije

Na osnovu Vizingove teoreme mi već znamo da je minimum jedan od dva moguća rešenja, zato uzimamo da je funkcija cilja broj konflikata koji treba svesti na nulu i prethodno staviti inicijalno rešenje na Δ broj boja. Ukoliko ne uspemo da dodjemo do 0 povećavamo broj boja na Δ + 1. Grafovi su predstavljeni preko biblioteke networkx.

VNS

Implementiran je Variable Neighbourhood Search sa različitim lokalnim pretragama i shakingzima.
Rešenje se čuva kao lista boja. Prvobitno rešenje uzima |E| broj boja iz opsega [1, Δ] slučajnim izborom.

Ivice se čuvaju kao parovi čvorova. Funkcija NormEdges daje da se parovi uvek čuvaju sortirani rastuće. Takodje, za lakši rad, čuvaju se i strukture podataka sa indkeksiranim ivicama i za svaku ivicu lista njenih suseda.

Prvo se nalazi čvor sa najvećim stepenom i ta vrednost se uzima za Δ. Zatim se nalaze komšije za svaku ivicu. Svaki konflikt se prvobitno broji dva puta, pa se konačna vrednost dobija deljenjem sa dva. Radi bržeg rada, ponovno brojanje konflikata je olakšano tako što se računa da li sa novom bojom imamo konflikt manje ili smo pridodali jedan.

Local searches:

1) **First improvement**

   Da ne bi bilo pristrasnosti, indeksi rešenja se “mešaju”. Za svaki indeks se prolazi kroz sve boje, ista se preskače. Ukoliko je sa novom bojom fitness manji, prekidamo pretragu. Ukoliko ne, vraćamo na staru boju.

2) **Best improvement**
   Za svaki indeks rešenja, prolazi se kroz sve boje i pamti se najbolji fintess kao i indeks sa najboljom novom bojom. Nakon svake provere vraćamo se na staru boju. Ukoliko smo našli bolje rešenje, na zapamćeno mesto stavljamo zapamćenu boju.

3) **Tabu search**, na osnovu drugih radova uvidjeno je da VNS zajedno sa tabu search daje odlične rezultate (Tabucol za vertex coloring)

Čuvamo rečnik zabranjenih poteza. Tražimo najbolji dozvoljen potez u datoj iteraciji. Za svaki indeks rešenja, prolazi se kroz sve boje i gleda se trenutno rešenje kao i najbolje rešenje. Čak iako je potez zabranjen, tabu, ako je fitness bolji od najboljeg dosadašnjeg pamtimo ga. Ako smo našli najbolji potez u iteraciji, zapravo ga menjamo. Stavljamo zabranu na vraćanje stare boje za odredjeni broj iteracija. Izlazak iz petlje je dostignut maksimum iteracija ili pronadjen fitness = 0.

Shakings:

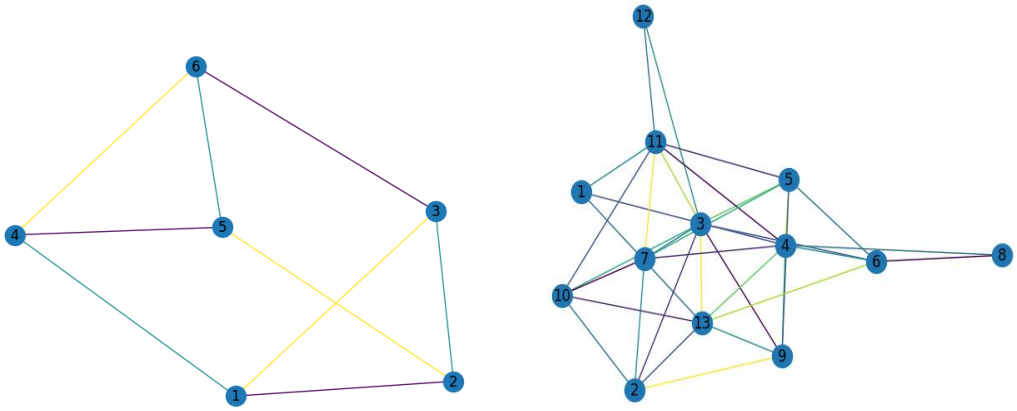
1) **K radnom** - biranje k random boja i menjanje istih u drugu random boju
2) **K swap** - k puta permutovati trenutno resenje
3) **Neighbours** - uzimanje random ivice i menjanje boje njenim susedima
4) **Incident (Conflict) edges** - od svih incidentnih ivica uzeti k random i promeniti boju

Napravljeno je 12 kombinacija rada VNS algoritma. U zavisnosti veličine i stepena grafa svaka kombinacija se pokazala drugačije. Takodje, u zavisnosti od vrste grafa, menjani su i VNS parametri. Vremensko ograničenje, za svaku komibinaciju, je 60s.

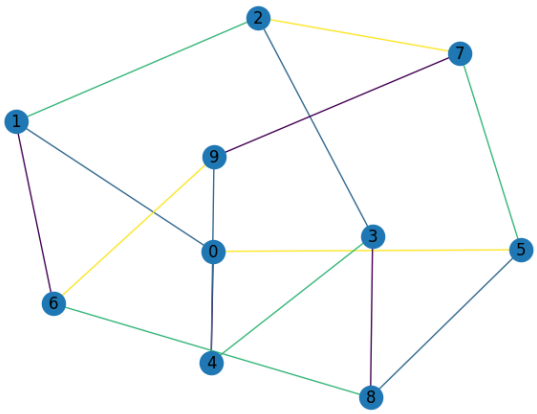
Rezultati su podeljeni po vrsti grafa, za svaku kombinaciju.

Rezultat = $\Sigma\left(\frac{konflikti=0}{ukupan\ broj\ grafova}\right) \times 100 - prosečno\ vreme$

- Mali grafovi – sve kombinacije dolaze do 0 konflikata u malom vremenskom periodu
  [k = 3, prob = 0.5, prostor pretrage je mali, ubrzavanje istraživanja prostora rešenja]

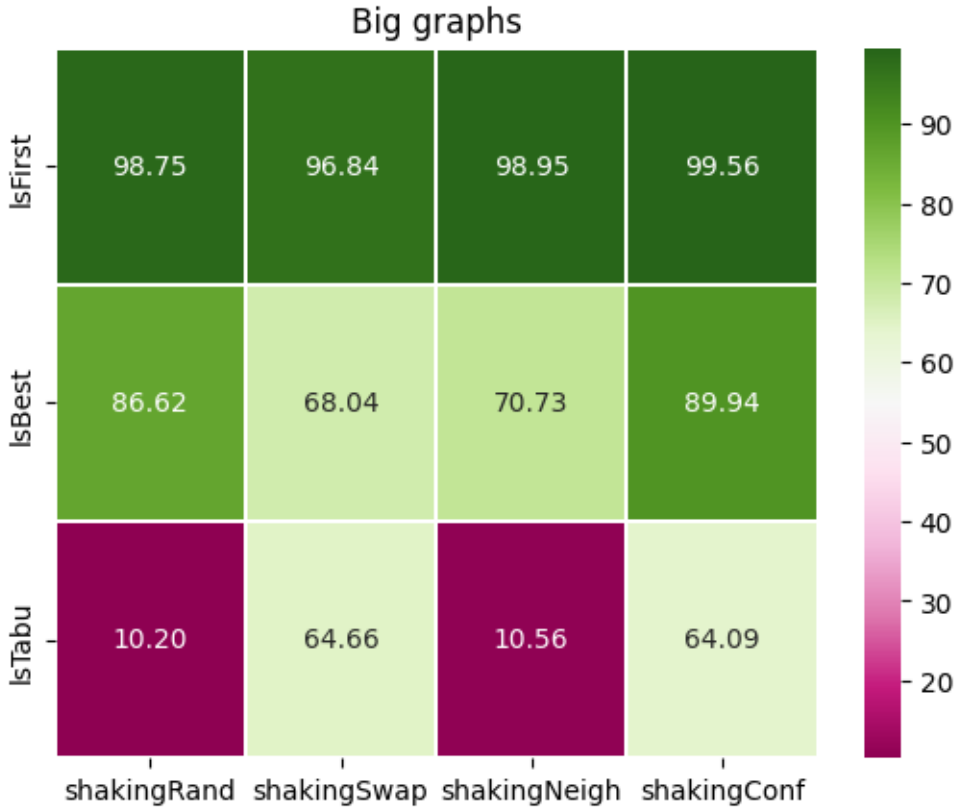


- SNARK grafovi, grafovi malog stepena ali ne mogu biti obojeni sa Δ boja. Kada se hromatski indeks poveća na Δ+1, rešenje se nalazi za manje od jedne sekunde.
  [k = 7, prob = 0.2, jači k zbog komplikovanije strukture grafa]

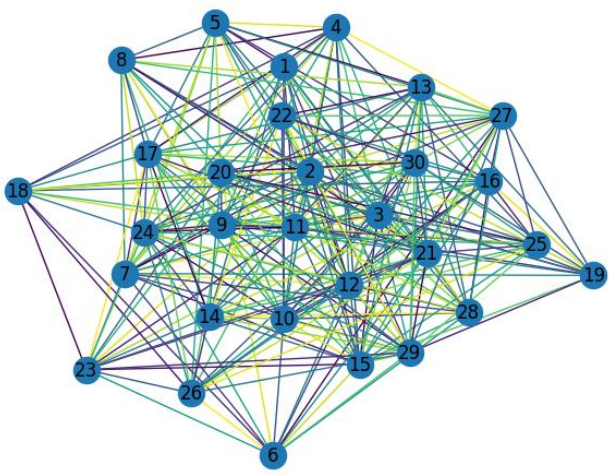


- **Veliki grafovi**(22 – 755 ivica)

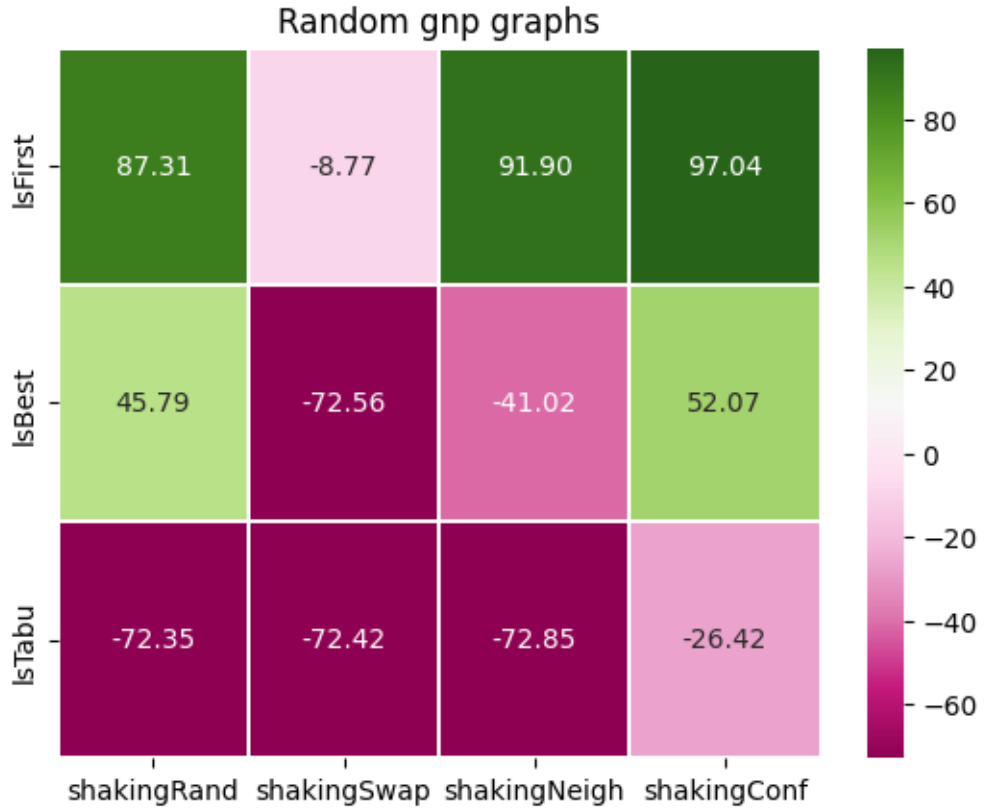
Neki su preuzeti iz graph color skupa instanci, neki su izgenerisani sa nx.gnm_random_graph(n,m) gde su n broj čvorova i m broj ivica.
[k = 6, prob = 0.2, drži se mala verovatnoća jer su grafovi osetljiviji i na malu promenu rešenja]



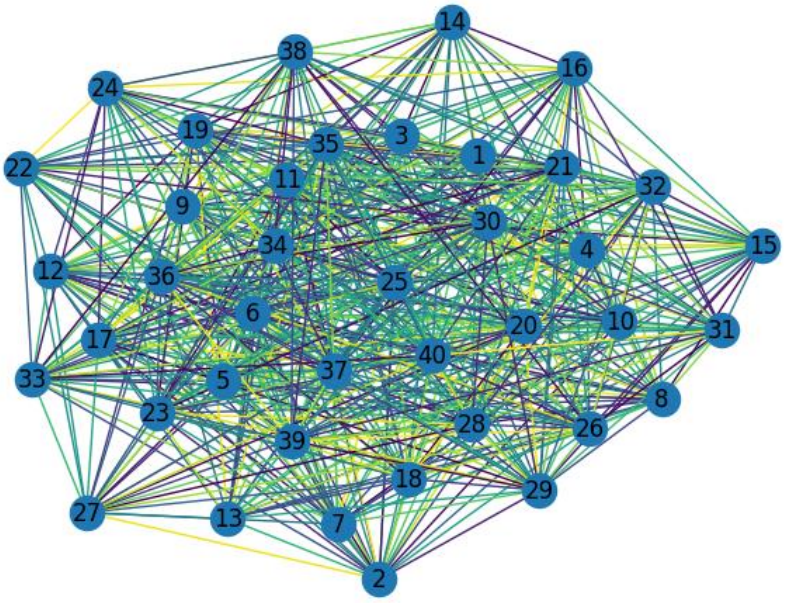
LsFirst se pokazuje kao najbolji izbor za lokalnu pretragu. LsBest je takodje ostario dobar učinak.
Uz shaking random i neighbours, lsTabu nije doneo dobre rezultate.



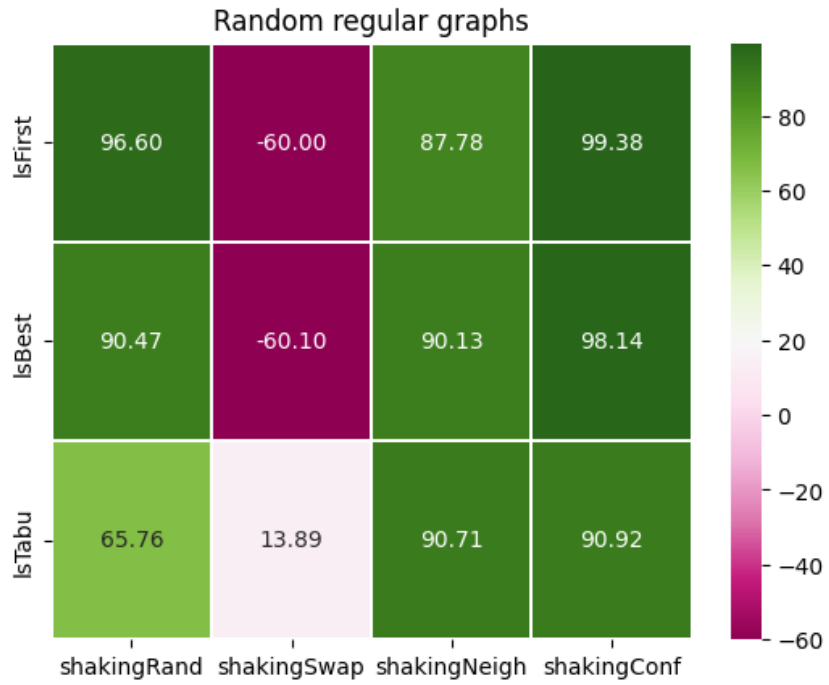
- **RandomGNP** - random generisan graf sa nx.gnp_random_graph(n,p) -gde su n broj čvorova i p verovatnoća za nastanak ivice.
  [k = 4, prob = 0.1]



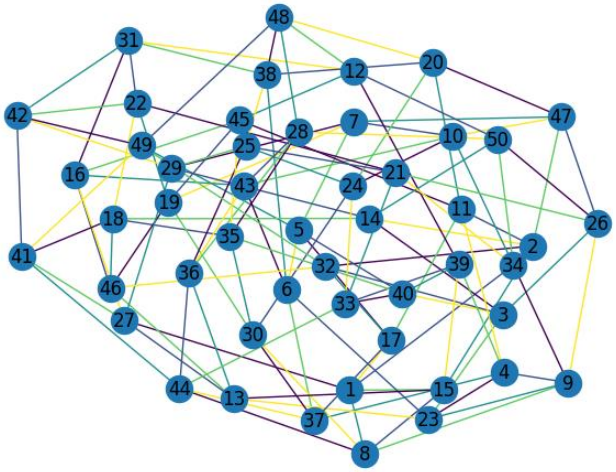
LsTabu ubedljivo najgori od lokalnih pretraga, dok je lsFirst najbolji. U ovom skupu grafova shaking swap pokazuje se kao loš pristup.



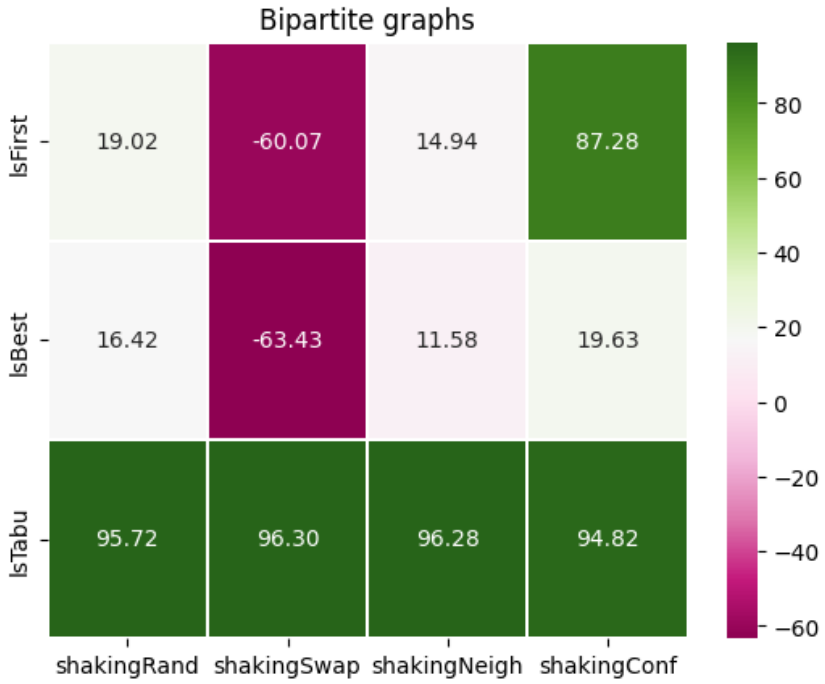
- **Random regular** – Regularni grafovi, svaki čvor ima isti broj ivica.
  Svi su izgenerisani sa nx.random_regular_graph(d, n), gde je d stepen svakog čvora i n broj čvorova.
  [k = 10, prob = 0.3, zbog homogenosti prostora rešenja k je velik]



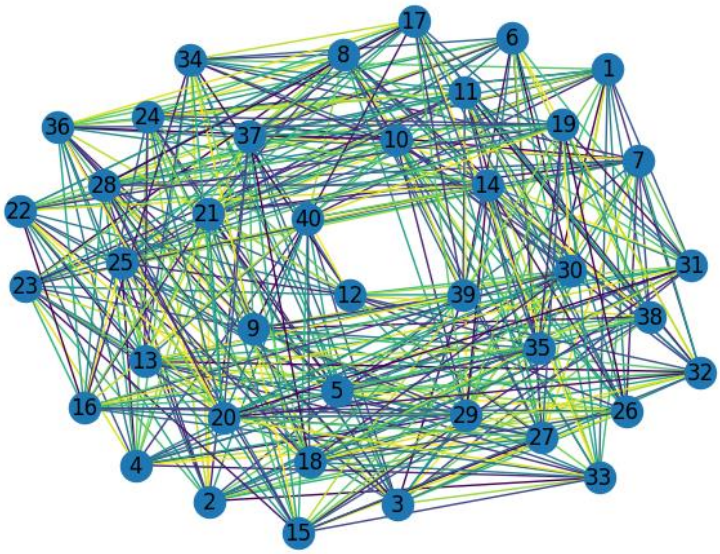
Nijedna lokalna pretraga uz shaking Swap ne daje dobre rezultate.
Sve tri lokalne pretrage su se odlično pokazale. Uz mali izuzetak lsTabu + shaking Random.



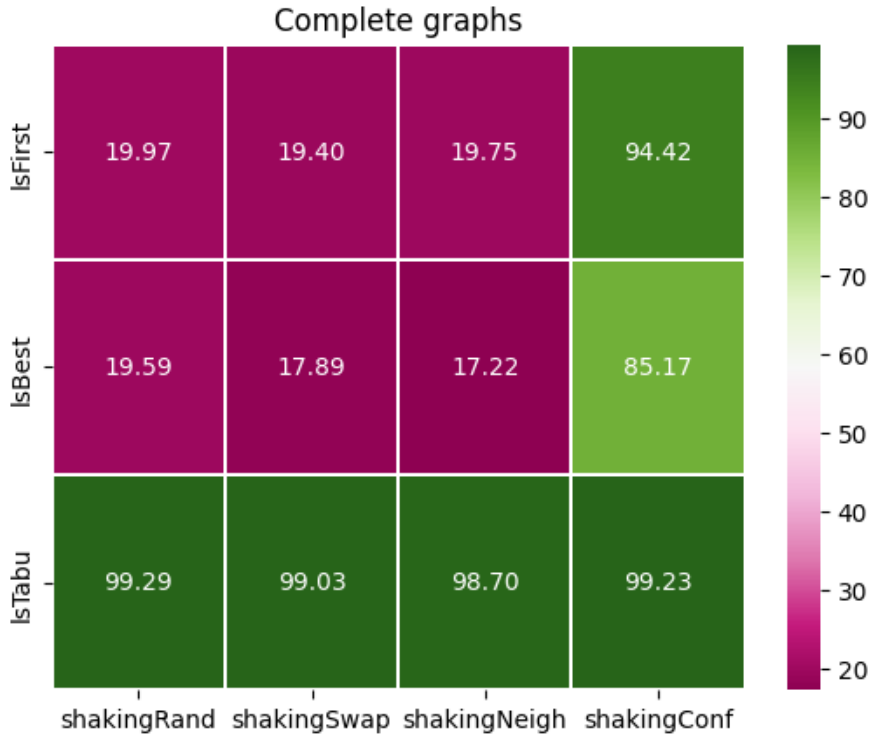
- **Bipartitivni** - Čvorovi su podeljeni u dva skupa, nijedna ivica ne spaja dva čvora iz istog skupa. Broj boja ivica mora biti Δ.
  Svi su izgenerisani sa nx.complete_bipartite_graph(n1, n2), gde su n1 i n2 gornje granice skupova čvorova.
  [k = 4, prob = 0.1, zbog simetričnosti grafa nema potrebe za jačim vrednostima]



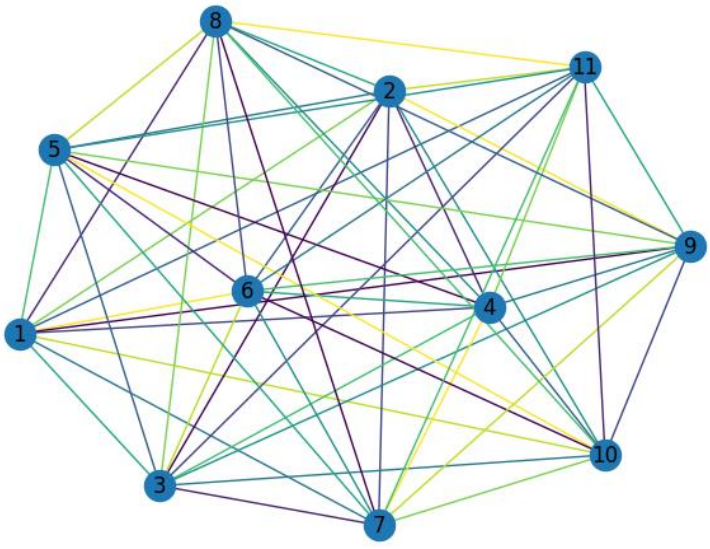
Za ovu vrstu grafova, lsTabu je ozbiljno bolji od ostale dve lokalne pretrage. Shaking swap ostaje kao loš shaking.



- **Complete** – kompletni grafovi, svaki čvor je direktno povezan sa ostalim čvorovima grafa. Teoreme kažu da ukoliko je broj čvorova paran, $\chi' = \Delta = n - 1$, a ako je neparan $\chi' = \Delta + 1 = n$
  Svi su izgenerisani sa nx.complete_graph(n), gde je n broj čvorova. [k = 4, prob = 0.1, zbog simetričnosti grafa nema potrebe za jačim vrednostima]



LsTabu ubedljivo najbolja lokalna pretraga. Shaking conflicts, bez obzira sa kojom lokalnom pretragom daje dobre rezultate.



- Izdvojeni slučajevi

  Za razliku od lsFirst, lsTabu i lsBest nisu uspeli da nadju rešenje sa 0 konflikata u zadatom vremenu. Štaviše, rešenje su našli prolaskom kroz samo jednu okolinu. Parametar vremena nije promenjen jer je ipak moguće naći rešenje za 60s.

[k = 12, prob = 0.2, grafovi su izuzetno gusti i potreban je veći broj okolina, verovatnoća se drži na maloj vrednosti radi osetljivosti pretrage]

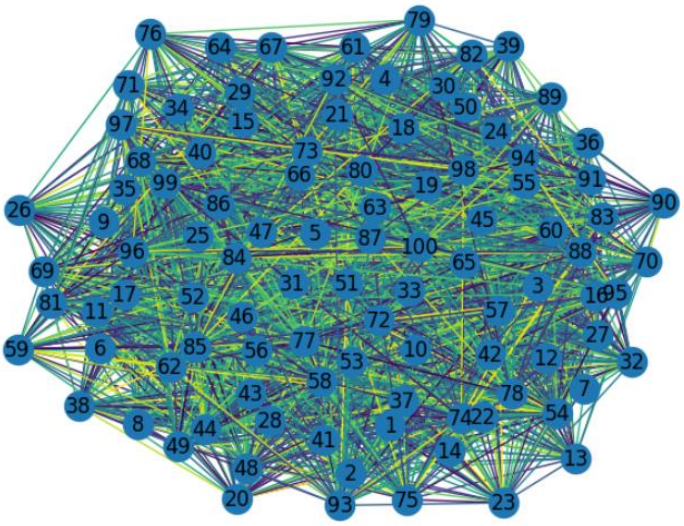
Gcol graf - preuzet iz graph color skupa instanci, 100 čvorova i 2487 ivica

Graph can be coloured with 61 or 62 colours. Class 1 or class 2 respectively.

	localSearch	shaking	num of conf/fitness	num of colours	time
0	lsFirst	shakingRand	0	61	35.3623
1	lsFirst	shakingSwap	0	61	25.7025
2	lsFirst	shakingNeigh	1	61	88.8330
3	lsFirst	shakingConf	0	61	6.79852
4	lsBest	shakingRand	3	61	954.186
5	lsBest	shakingSwap	4	61	974.933
6	lsBest	shakingNeigh	5	61	987.796
7	lsBest	shakingConf	5	61	1020.34
8	lsTabu	shakingRand	637	61	541.768
9	lsTabu	shakingSwap	594	61	568.764
10	lsTabu	shakingNeigh	582	61	538.686
11	lsTabu	shakingConf	635	61	508.554

Graph is class 1, meaning chromatix index is equal to the graph's degree.

LsFirst i shaking conflicts zajedno daju odličan rezultat.

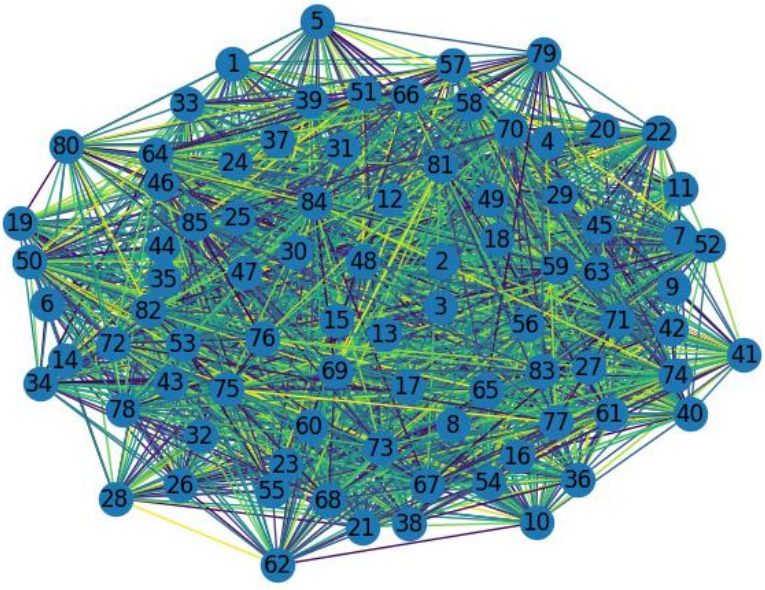


RandomGNP - 85 čvorova, 2129 ivica

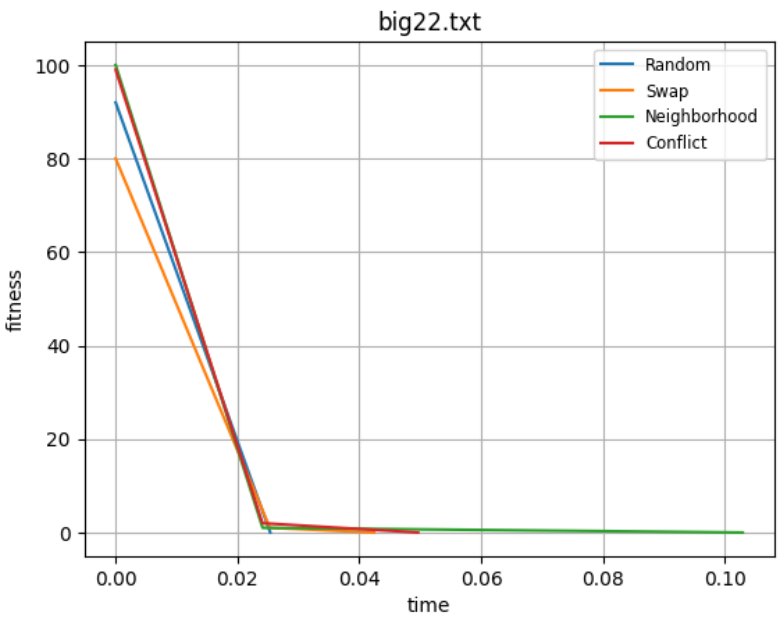
Graph can be coloured with 62 or 63 colours. Class 1 or class 2 respectively.

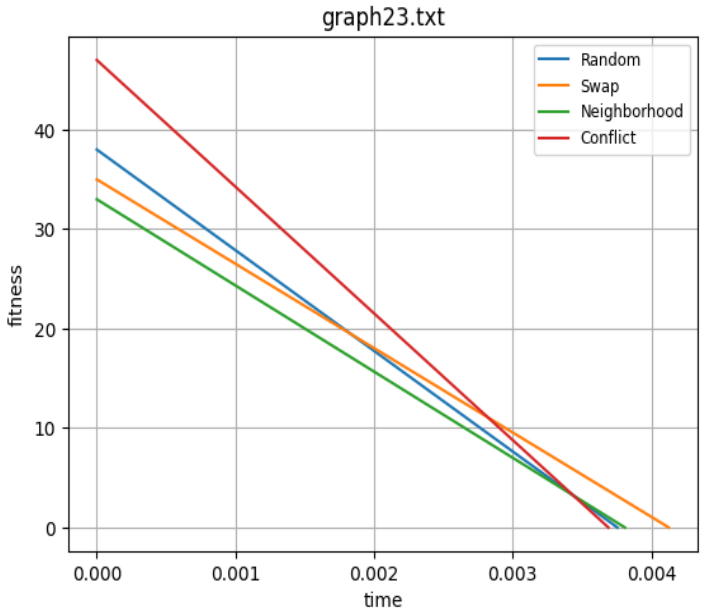
	localSearch	shaking	num of conf/fitness	num of colours	time
0	lsFirst	shakingRand	0	62	27.686
1	lsFirst	shakingSwap	0	62	16.2658
2	lsFirst	shakingNeigh	0	62	14.8318
3	lsFirst	shakingConf	0	62	5.05301
4	lsBest	shakingRand	1	62	1230.21
5	lsBest	shakingSwap	2	62	1325.55
6	lsBest	shakingNeigh	2	62	1264.31
7	lsBest	shakingConf	0	62	1226.13
8	lsTabu	shakingRand	382	62	714.605
9	lsTabu	shakingSwap	395	62	716.132
10	lsTabu	shakingNeigh	374	62	716.588
11	lsTabu	shakingConf	384	62	715.791

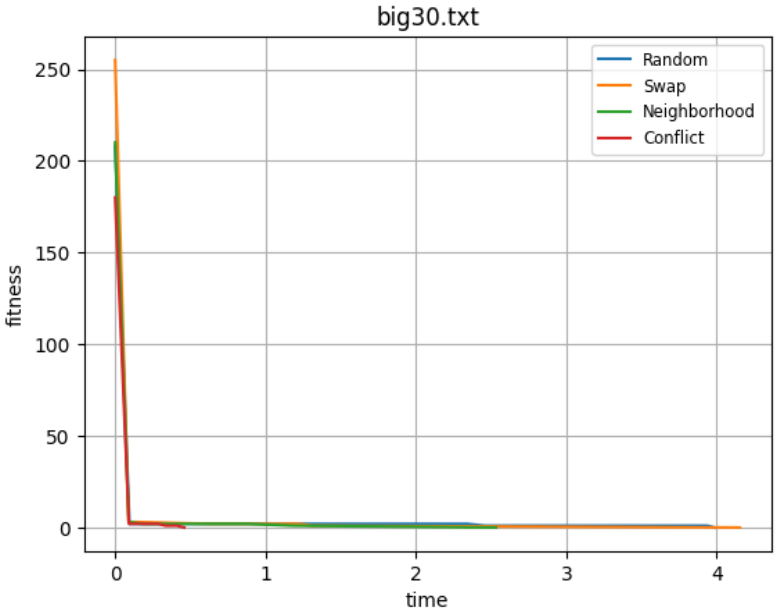
Graph is class 1, meaning chromatix index is equal to the graph's degree.

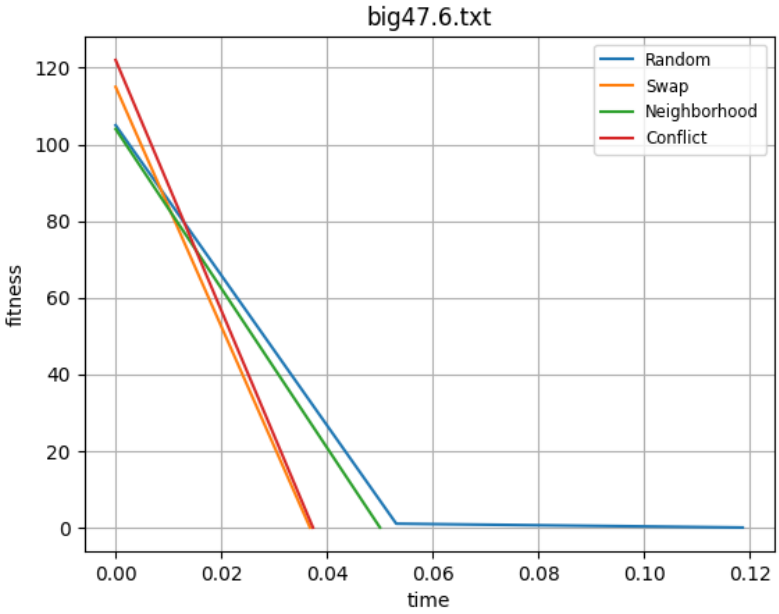


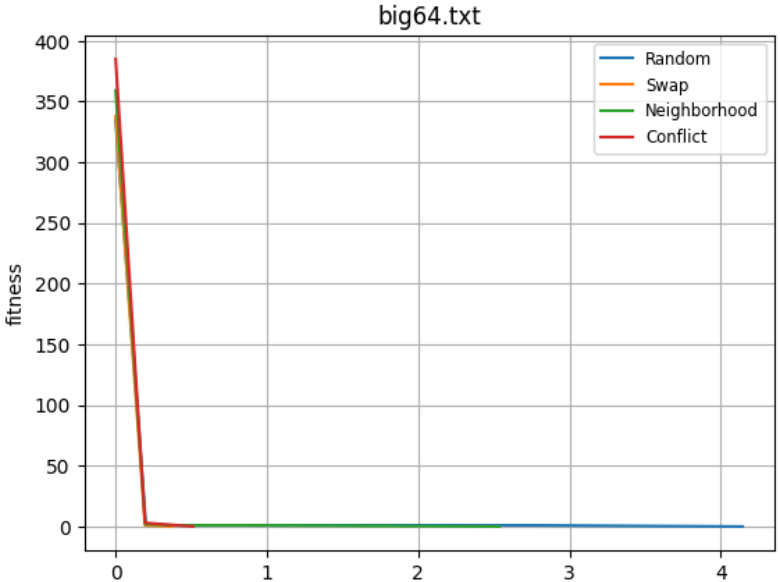
Konvergencija fitness-a u toku izvršavanja za fiksiranu lokalnu pretragu lsFirst.

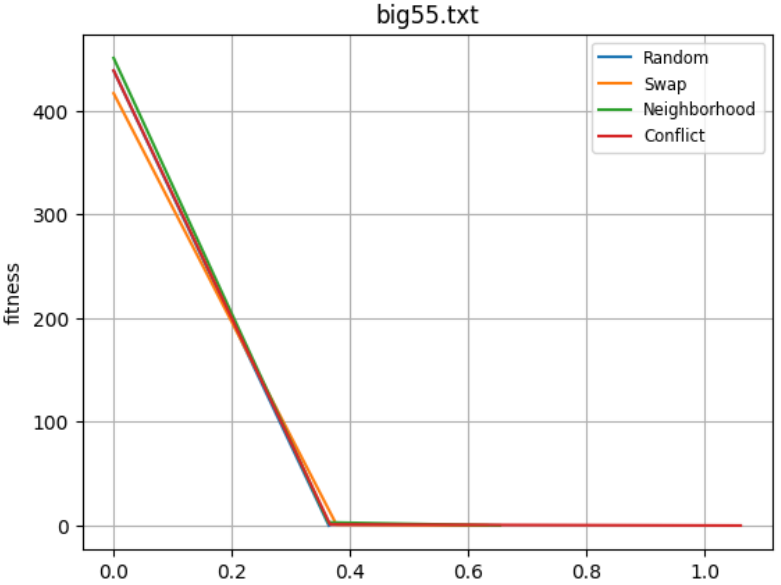


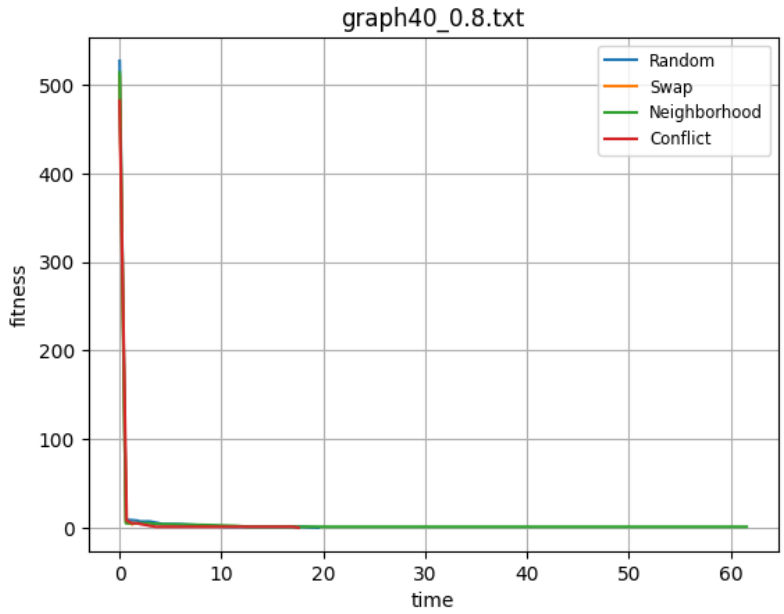


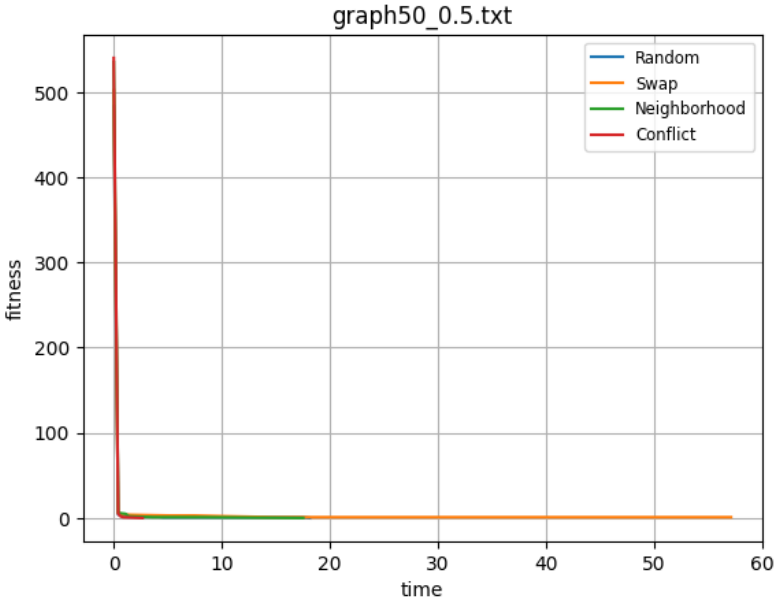


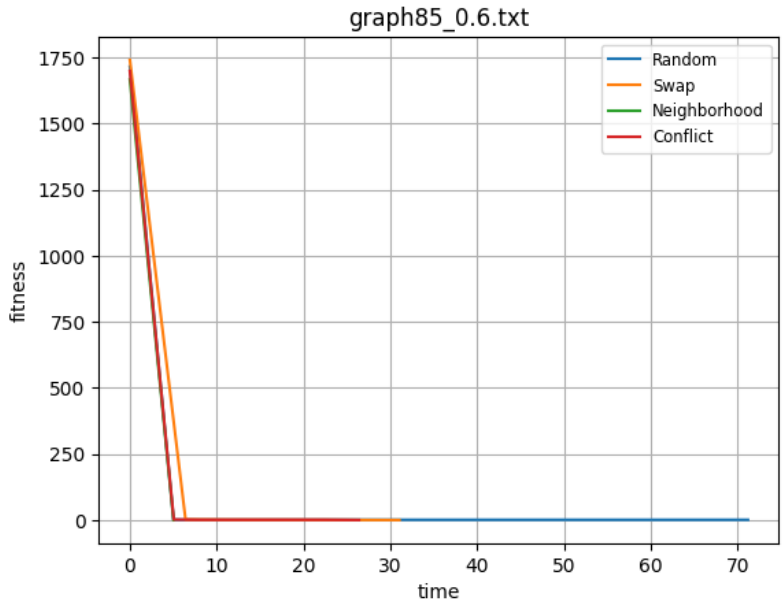


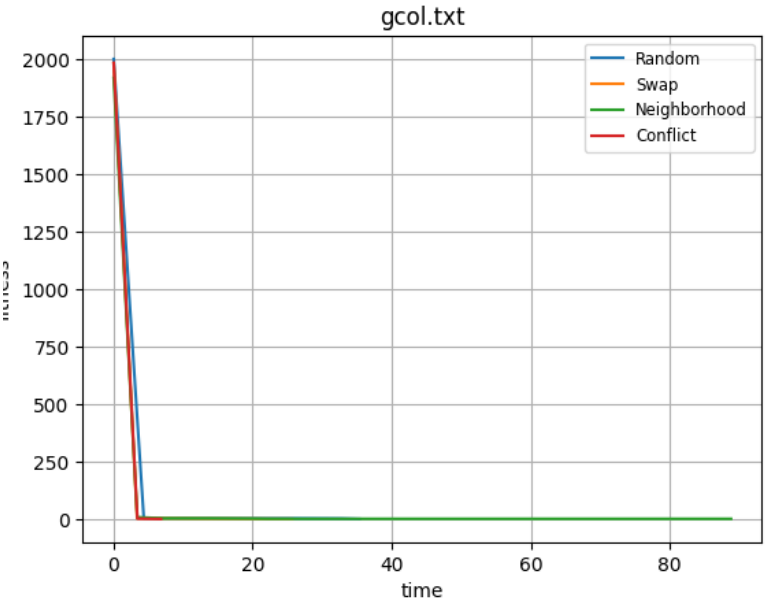





graph40_0.8.txt
Random
Swap
Neighborhood
Conflict
fitness
time
0
100
200
300
400
500
0
10
20
30
40
50
60


graph50_0.5.txt
Random
Swap
Neighborhood
Conflict
fitness
time
0
100
200
300
400
500
0
10
20
30
40
50
60


graph85_0.6.txt
Random
Swap
Neighborhood
Conflict
fitness
time
0
250
500
750
1000
1250
1500
1750
0
10
20
30
40
50
60
70


gcol.txt
Random
Swap
Neighborhood
Conflict
time
0
250
500
750
1000
1250
1500
1750
2000
0
20
40
60
80

Zaključak:
Gledajući opšte grafove(bigGraphs, randomGNP) lsTabu, iako dobro ocenjen kod bojenja čvorova, ovde se nije toliko dobro pokazao. Za razliku od lsFirst koji se izdvaja kao najbolja opcija za lokalnu pretragu.
Shaking swap se pokazuje kao nestabilna opcija diverzifikacije. Kod malih grafova se čak izdvaja kao jedan od najboljih (u zavisnosti od pretrage), dok kod drugih ne uspeva da dodje do rešenja i daje loš generalni utisak.
Za teoremske grafove, gde obično znamo broj boja, lsTabu je uspevao da dodje do rešenja, za njim i lsFirst.
LsBest u obe grupe nije dolazio do izražaja jer su ostale dve pretrage imale jače oscilacije. Svakako, pokazao se kao pouzdan izbor za grafove ispod 1000 ivica.
Shaking Conflicts se uvek provlačio kao odlična opcija, što je i očekivano jer iako ima random deo ipak direktnije pristupa rešavanju konflikata.

Testiranje je vršeno na laptopu sa 13th Gen Intel Core™ i7-13620H × 16 procesorom, pod operativnim sistemom Ubuntu 24.04.3 LTS.

GA

U fokusu ovog poglavlja je implementacija genetskog algoritma, metaheurističkog pristupa koji simulacijom prirodne selekcije rešava problem bojenja ivica grafa, **težeći ka eliminaciji svih konflikata boja na susednim granama kroz evolutivni razvoj rešenja.**

Jedna od ključnih karakteristika ove implementacije je odstupanje od tradicionalnog objektno-orijentisanog pristupa koji podrazumeva definisanje posebne klase za svaku jedinku (npr. klasa Individual). Umesto toga, korišćene su **sirove strukture podataka** (liste i nizovi), što donosi nekoliko značajnih prednosti:

- **Smanjenje memorijskog zauzeća (Memory Overhead):** Izbegavanjem instanciranja hiljada objekata klase, memorija se koristi isključivo za čuvanje genetskog materijala.
- **Povećana brzina izvršavanja:** Direktna manipulacija nad listama i numpy nizovima omogućava brži pristup podacima i efikasnije izvršavanje genetskih operatora poput ukrštanja i mutacije.
- **Jednostavnost serijalizacije:** Podaci su u svakom trenutku u formatu pogodnom za matematičke operacije i logičku proveru konflikata.

Struktura hromozoma (Reprezentacija podataka)

U ovom algoritmu, hromozom nije definisan kao kompleksan objekat ili klasa, već kao **linearni niz (lista) celih brojeva**. Svaki hromozom predstavlja jedno kompletno potencijalno rešenje problema bojenja ivica.

Svaki element u listi naziva se **gen**.

- **Indeks gena:** Predstavlja redni broj ivice u grafu. Redosled je fiksiran na početku programa kako bi svaki hromozom bio konzistentan.
- **Vrednost gena:** Predstavlja boju (identifikator boje) koja je dodeljena toj ivici.

U funkciji initialPopulation, hromozom se kreira na sledeći način:

individual = [random.randrange(maxNumOfColours) for _ in range(numEdges)][1]

[1] Iako smo ovde dodali kod, i dodavacemo mozda na jos nekim mestima, vazno je naglasiti da se u nastavku materijala nece svuda isticati objasnjenje u vidu koda kao reprezentacija, ceo kod se moze pogledati u .ipynb fajlovima

Ovakva struktura omogućava direktnu komunikaciju sa networkx bibliotekom. Kada želimo da proverimo fitness, mi zapravo prolazimo kroz ovu listu i upoređujemo vrednosti na osnovu povezanosti ivica koju nam diktira struktura grafa.

Inicijalizacija populacije (Početak evolucije)

Nakon što smo definisali strukturu hromozoma, sledeći korak je kreiranje **početne populacije**. U genetskim algoritmima, ovaj korak je presudan jer obezbeđuje raznolikost iz kojeg će algoritam kasnije birati najbolje elemente.

Umesto da pokušavamo da pogodimo tačno bojenje, mi generišemo popSize potpuno nasumičnih rešenja. To radimo koristeći funkciju initialPopulation.

Logika u kodu: Za svaku jedinku u populaciji, prolazimo kroz sve ivice grafa i svakoj dodeljujemo nasumičnu vrednost (boju) iz opsega [0,maxNumOfColours−1].

Fitnes funkcija (Evaluacija rešenja)

Fitnes funkcija u ovom radu dizajnirana je tako da na veoma efikasan način detektuje konflikte u bojenju. Umesto standardnog (i sporog) poređenja svakog para ivica, primenjen je pristup **grupisanja po čvorovima (Node Slots)**.

Kako fitnes zapravo radi (Logika koda):

1. **Mapiranje boja na čvorove:** Za svaki čvor u grafu kreiramo listu (node_slots). Prolazimo kroz sve ivice i boju svake ivice "upisujemo" u oba čvora koja ta ivica spaja.
2. **Grupisanje konflikata:** Ako se u listi jednog čvora ista boja pojavi više puta, to znači da taj čvor ima više ivica iste boje — što je direktan konflikt.
3. **Matematički obračun kazni:** Da bi precizno izračunala koliko ima konflikata, koristimo sortiranje i kombinatornu formulu $\frac{n\cdot(n-1)}{2}$ (gde je n broj ivica iste boje u jednom čvoru). Ovo osigurava da se svaka kombinacija pogrešno obojenih ivica kazni.

Elitizam (Očuvanje najboljih rešenja)

Nakon što se izvrši evaluacija celokupne populacije pomoću fitnes funkcije, primenjuje se mehanizam **elitizma**. U ovoj implementaciji, elitizam je postavljen tako da se **dve najbolje jedinke** (one sa najmanjim brojem konflikata) direktno prebacuju u narednu generaciju.

U funkciji runGaModular, ovaj proces se obavlja u dva koraka:

1. **Rangiranje:** Populacija se sortira prema vrednosti fitnesa.
2. **Direktno kopiranje:** Prva dva hromozoma sa vrha liste se uzimaju bez ikakvih izmena i čine bazu nove populacije (newPop).

Zašto baš dve? Čuvanjem dve najbolje jedinke osiguravamo da se trenutni "rekord" u broju konflikata ne izgubi, dok istovremeno ostavljamo dovoljno mesta u ostatku populacije (npr. ostalih 98 mesta) za istraživanje novih kombinacija boja kroz selekciju, ukrštanje i mutaciju.

Selekcija roditelja (Selection)

Nakon očuvanja elite, ključni korak je odabir jedinki koje će postati roditelji nove generacije. Suština selekcije je u balansu između dva faktora: **iskorišćavanja** trenutno najboljih rešenja i **istraživanja** novih prostora pretrage. U radu su implementirane tri strategije koje na različite načine upravljaju ovim balansom:

1. **Turnirska selekcija (selectionTournament)**:Simulacija prirodne kompeticije u malim grupama. Izborom najboljeg od tri nasumična kandidata, algoritam favorizuje kvalitet, ali dozvoljava i prosečnim jedinkama da pobede ako se nađu u "slabijoj grupi".Pruža najjači selektivni pritisak i najbrže vodi ka rešenju.
2. **Ruletska selekcija (selectionRoulette)**: Verovatnoća po principu "zasluge". Budući da minimizujemo broj konflikata, svaka jedinka dobija šansu srazmernu vrednosti 1/f. Obezbeđuje visok genetski diverzitet (raznolikost), jer čak i lošija rešenja zadržavaju šansu da prenesu svoje specifične gene koji se kasnije mogu pokazati korisnim.
3. **Rang selekcija (selectionRank)**:Standardizacija šanse na osnovu pozicije. Umesto da gleda koliko je neko bolji, gleda samo *da li je* bolji (prvi na listi je uvek najbolji, bez obzira na razliku u bodovima).Sprečava preranu konvergenciju u situacijama kada jedna jedinka dominira populacijom, čime se izbegava brzo upadanje u lokalni optimum.

Ukrštanje (Crossover)

Ukrštanje je ključni operator genetskog algoritma koji omogućava kombinovanje dva roditelja kako bi se stvorio potomak. Cilj je da potomak nasledi najbolje karakteristike (boje ivica) od oba roditelja. U kodu su implementirane tri varijante ovog procesa:

1. Ukrštanje u jednoj tački (crossoverSinglePoint): Ovo je najjednostavnija metoda gde se nasumično bira jedna tačka preseka u hromozomu.Prvi deo niza boja uzima se od jednog roditelja, a ostatak od drugog.Ova metoda čuva veće blokove (sekvence) boja, što je korisno ako roditelji već imaju dobro obojene čitave delove grafa.

2. Ukrštanje u dve tačke (crossoverTwoPoint): Naprednija verzija gde se biraju dve tačke preseka, čime se hromozom deli na tri segmenta.Potomak dobija početak i kraj od prvog roditelja, dok se središnji segment preuzima od drugog.Pruža veću fleksibilnost u kombinovanju i omogućava "umetanje" dobrih genetskih blokova iz sredine hromozoma, što smanjuje rizik od preteranog očuvanja samo krajnjih sekvenci.

3. Uniformno ukrštanje (crossoverUniform): Ovaj pristup potpuno napušta koncept tačaka preseka.Za svaku pojedinačnu ivicu u grafu, baca se novčić (verovatnoća 0.5) i odlučuje se da li će boja biti uzeta od prvog ili drugog roditelja.Ovo je najagresivniji vid ukrštanja. Maksimalno meša gene i odličan je za razbijanje lokalnih optimuma, jer se boje ivica tretiraju potpuno nezavisno.

- **Single-point** je stabilniji i polako gradi rešenje.
- **Uniform** brže istražuje nove kombinacije, ali može biti "razoran" za već skoro savršena rešenja.

Mutacija (Mutation)

Mutacija je operator koji uvodi male, nasumične promene u genetski kod jedinki. Njena glavna uloga je sprečavanje **lokalne konvergencije** – situacije u kojoj svi hromozomi postanu previše slični, pa algoritam prestane da istražuje nova rešenja. U radu su implementirana tri nivoa mutacije:

1. Nasumična mutacija (mutationRandom): Ovo je osnovni oblik mutacije.Potpuno nasumično bira jednu ivicu u grafu i dodeljuje joj bilo koju boju iz dozvoljenog opsega. Održava genetski diverzitet "na slepo", osiguravajući da prostor pretrage ostane otvoren.

2. Pametna mutacija (mutationSmart): Napredniji pristup koji pokušava da izbegne konflikte u jednom koraku. Kada izabere ivicu za mutaciju, algoritam prvo pogleda koje su boje već zauzete na njenim krajevima (susedne ivice). Zatim pokušava da izabere boju koja nije u tom skupu zabranjenih boja. Direktno smanjuje broj konflikata već u samom procesu mutacije, delujući kao lokalna optimizacija.

3. Mutacija zasnovana na konfliktu (mutationConflict): Algoritam prvo skenira ceo graf i identifikuje isključivo one ivice koje trenutno učestvuju u konfliktu (koje imaju istu boju kao njihovi susedi). Umesto da menja bilo šta, on bira jednu od tih "problematičnih" ivica i menja joj boju. Fokusira resurse algoritma na kritične tačke. Ako je 90% grafa dobro obojeno, ova mutacija neće kvariti ono što je dobro, već će uporno pokušavati da popravi onih 10% koji kvare fitnes rezultat.

Algoritam je dizajniran modularno, što omogućava testiranje različitih evolutivnih strategija. Za svaki graf su pokrenuti testovi sa različitim kombinacijama operatora kako bi se utvrdilo koja kombinacija najbrže dolazi do optimalnog rešenja (hromozoma sa fitnesom 0). Ukupno je pokrenuto 27 kombinacija svih operatora (selekcije, ukrstanje i mutacija) i rezultati svih kombinacija prikazani su u resultsGAold i resultsGA. U okviru fajlova ga.ipynb i gaExperimentalLOCALSEARCH.ipynb se nalazi vizuelne prezentacije najbolje kombinacije od svih i kako su se top 3 najbolje kombinacije ponasale u grafiku konvergencije. Grafik konvergencije ukazuje na ponasanje fitnesa kroz povecanje generacija.

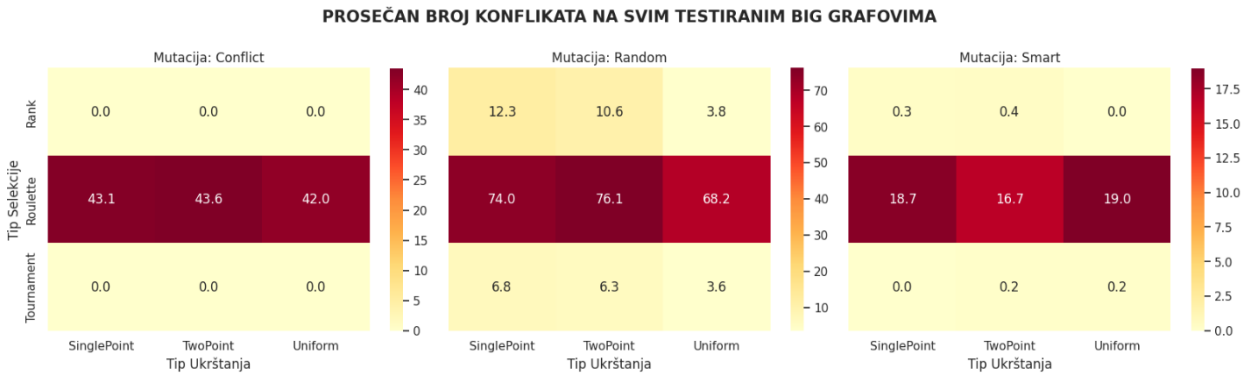
Spomenuto je da postoje dva foldera sa rezultatima, a to su resultsGAold i resultsGA. Zasto? Ono sto je vazno istaci je da na odredjenim test primerima nas genetski algoritam nije davao ocekivane rezultate koje nam i Vizingova teorema kaze, pa je bilo neophodno nadograditi ga i unaprediti sa lokalnom pretragom o kojoj cemo govoriti kasnije. Tako da resultsGAold folder predstavlaj rezultate koji su tu kao reprezentacija da nas genetski algoritam nije davao zadovoljavajuce rezultate na vecini eksperimentilanih grafova, mnogi od njih su teorijski grafovi za koje vaze odredjena pravila, a samim tim se i ocekuju odredjeni rezlutati.

Sada cemo prikazati za pocetak neke statitike i zapazanja za grafove koji su mogli da se rese sa “cistim” gentskim algoritmom, tu spadaju :

• Mali grafovi – sve kombinacije dolaze do 0 konflikata u malom vremenskom periodu, tako da tu nemamo nista specijalno da istaknemo.

• SNARK grafovi - grafovi malog stepena ali ne mogu biti obojeni sa Δ boja. Kada se hromatski indeks poveća na Δ+1, rešenje se nalazi za manje od jedne sekunde. (flowerSnark i petersen).

• Veliki grafovi(22 – 755 ivica) - Neki su preuzeti iz graph color skupa instanci, neki su izgenerisani sa nx.gnm_random_graph(n,m) gde su n broj čvorova i m broj ivica.



Posto heatmape su najzgodnije za prikaza u 2D, na svakoj od “grupa” grafova koje cemo prokomentarisati isaci cemo neke druge podatke. Ono na sta je stavljen akcenat kod velikih grafova su mutacije. Mozemo primetiti da je mutacija Conflict pokazala najbolje rezultate jer su u svim kombinacijama selekcija i ukrstanja rezultati na heatmapi 0.0. Medjutim tip selekcije Rulet u svim slucajevima daje najgore rezultate za velike grafove.

Sledece sto cemo obrazloziti je vec nesto sto smo napomenuli, a to je da je koriscena lokalna pretraga za mnoge eksperimentalne grafove kako bi se postigli zeljeni rezultati. Implementirane su dve varijante lokalne pretrage za popravku rešenja:

1. **LocalSearch:** Fokusira se na pronalaženje potpuno slobodne boje. Ukoliko su sve boje zauzete, uvodi nasumičnu promenu kako bi se stimulisao diverzitet.

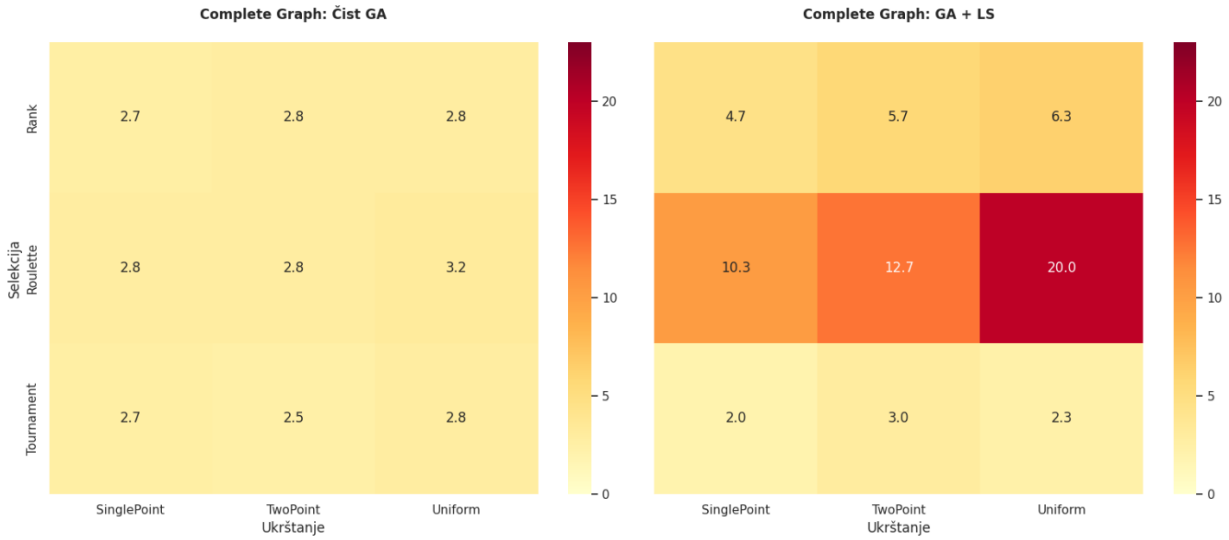
2. **GreedyLocalSearch :** Pored traženja slobodne boje, u kritičnim situacijama primenjuje *greedy* strategiju. Algoritam ispituje svaku boju i bira onu koja minimizuje broj lokalnih konflikata. Ova varijanta se pokazala superiornom kod grafova sa velikim stepenom čvorova gde je prostor za slobodne boje izuzetno ograničen.

Druga lokalna pretraga je primenjena na test primere gcol, graph85 i bipartitni20.

Takodje vazno je istaci da smo u ovom trenutku uveli i vremensko ogranicenje kako nam program ne bi trajao predugo i svaka kombinacija pokusala je da nadje resenje za neko odredjeno vreme koje mu zadamo. Ovo sprečava algoritam da troši prevelike resurse na ekstremno složenim grafovima koji zahtevaju više od predviđenog vremena za konvergenciju. U slučajevima prekida, rezultati se beleže do poslednje uspešne generacije, čime se omogućava analiza napretka čak i kod nedovršenih procesa.

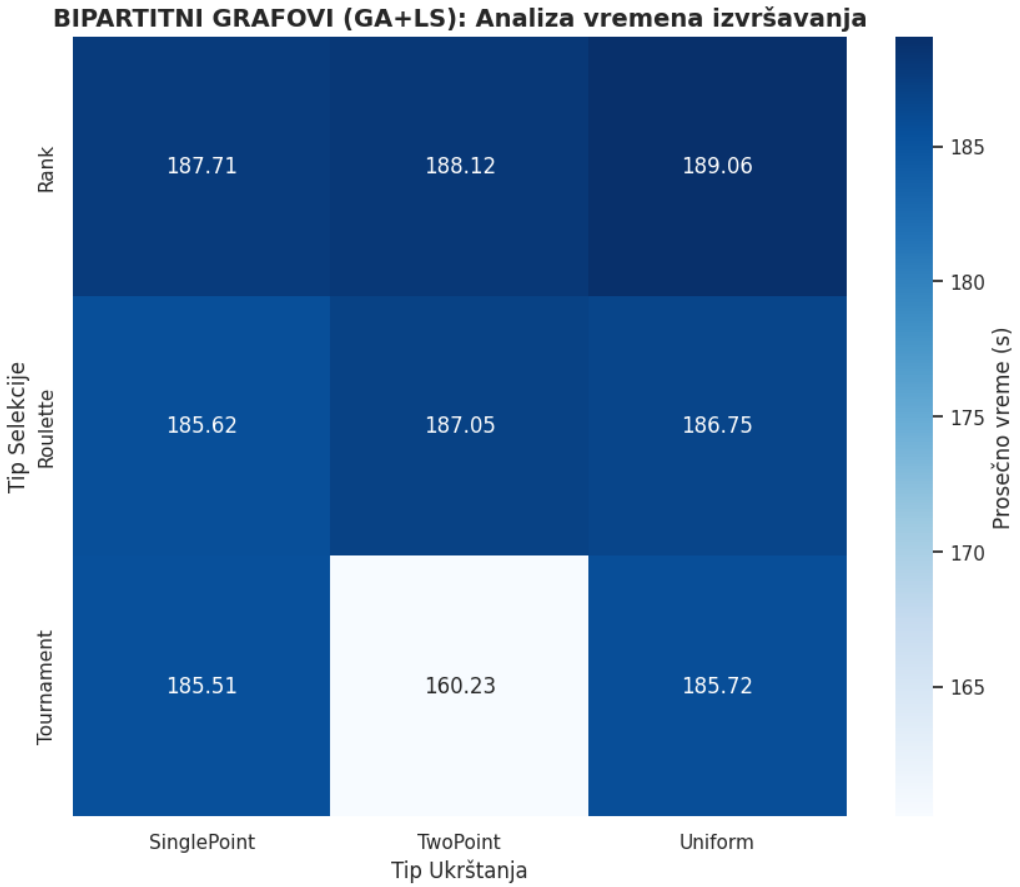
Ali pre nego sto prikazemo heat mape za eksperimentalne grafove koji su testirani iskljucivo pomocu GA+LS metode, prvo cemo uporediti kompletne grafove. Tacnije imamo

dva grafa, jedan je obojen sa delta+1 samo pomocu GA, a drugi je obojen sa delta ali sa GA+LS metodom.



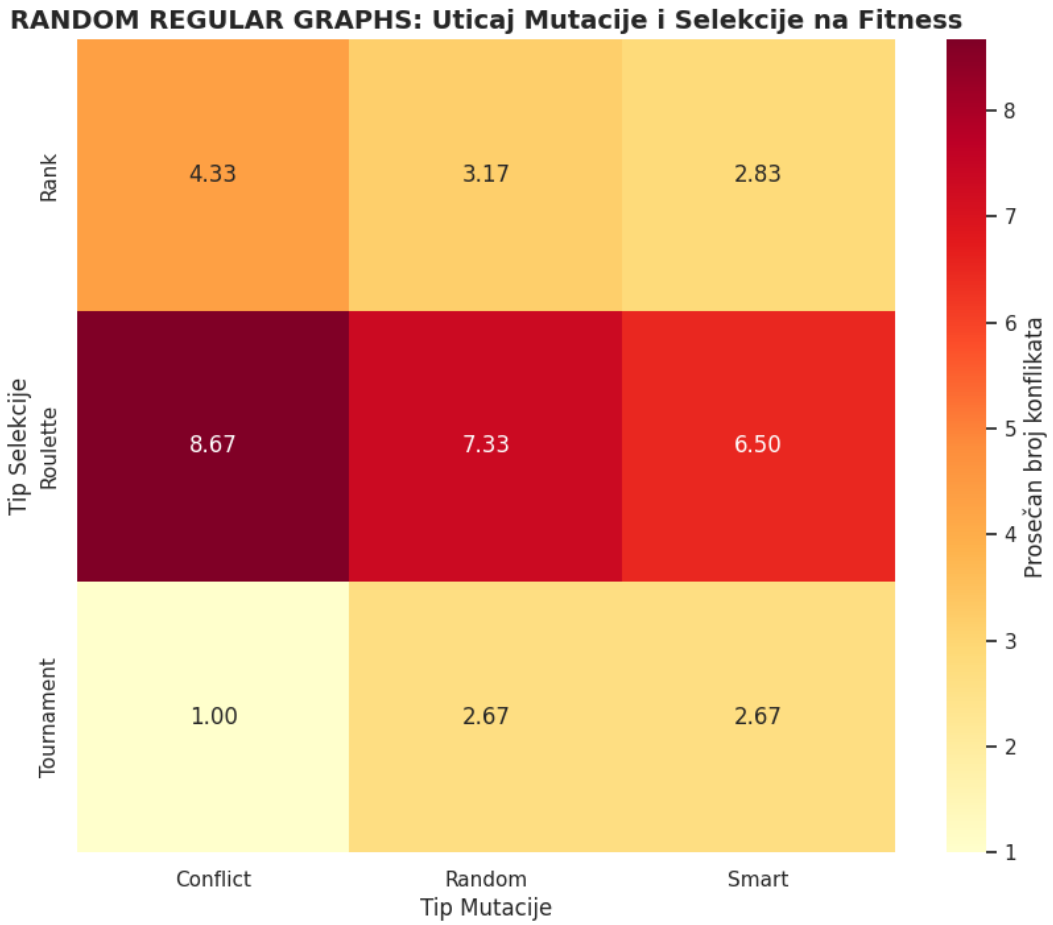
Ono sto ovde primecijemo kao interesantnu stvar, iako nije isti graf u pitanju i sa leve strane i desne strane. Sa leve strane imamo graf koji ima neparan broj cvorova i on se moze resiti po teoremi iskljucivo sa delta+1, ovaj graf ima dosta bolje rezultate i manji broj konflikta generalno, pogotovo je istaknuta razlika kada vidimo rulet selekciju. Tako da ne mora uvek znaciti da GA+LS je bolji od cistog GA. Nekada i cist ga moze dati izuzetne rezultate.

Sa narednom heatmapom cemo pokazati efikasnost (brzinu) naseg algoritma na specificnoj klasi grafova – bipartitinim grafovima. Bipartitivni - Čvorovi su podeljeni u dva skupa, nijedna ivica ne spaja dva čvora iz istog skupa. Broj boja ivica mora biti Δ.Svi su izgenerisani sa nx.complete_bipartite_graph(n1, n2), gde su n1 i n2 gornje granice skupova čvorova. Dok su prethodne mape koje smo analizirali pratile greške (fitness), ova plava mapa prati vreme.



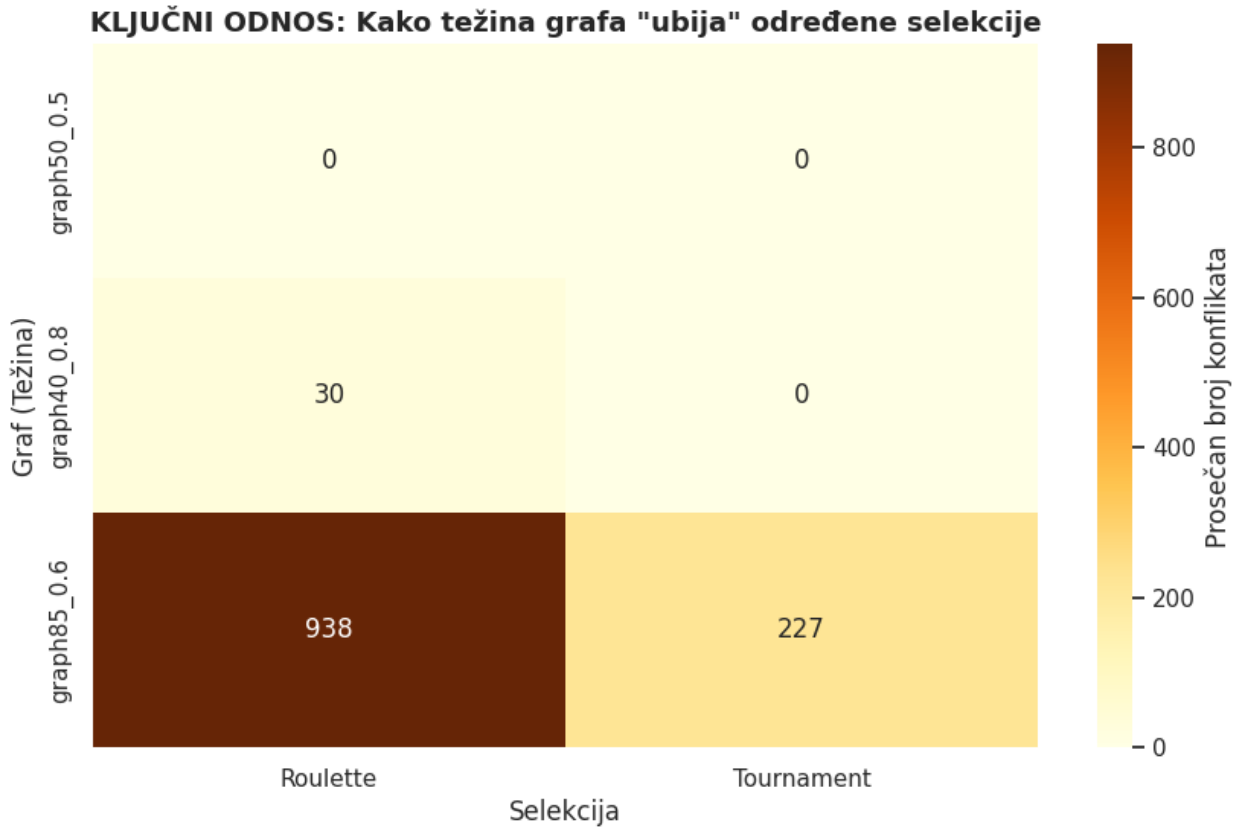
Za klasu bipartitnih grafova, primetno je da izbor operatora mutacije ima sekundarni značaj u poređenju sa brzinom konvergencije koju diktira lokalna pretraga. Budući da bipartitne strukture dozvoljavaju laku identifikaciju slobodnih boja, lokalna pretraga vrši korekcije brže i preciznije nego što bi to uradila bilo koja nasumična mutacija. U ovom slučaju, mutacija služi isključivo za održavanje minimalnog genetskog diverziteta, dok glavnu ulogu u pronalaženju rešenja preuzimaju operatori ukrštanja i proces lokalne optimizacije. Ono sto se takodje moze primetiti je da vreme u svim slucajevima je slicno i da se jedino istice kombincija Tournament + TwoPoint.

Random regular – Regularni grafovi, svaki čvor ima isti broj ivica. Svi su izgenerisani sa nx.random_regular_graph(d, n), gde je d stepen svakog čvora i n broj čvorova.



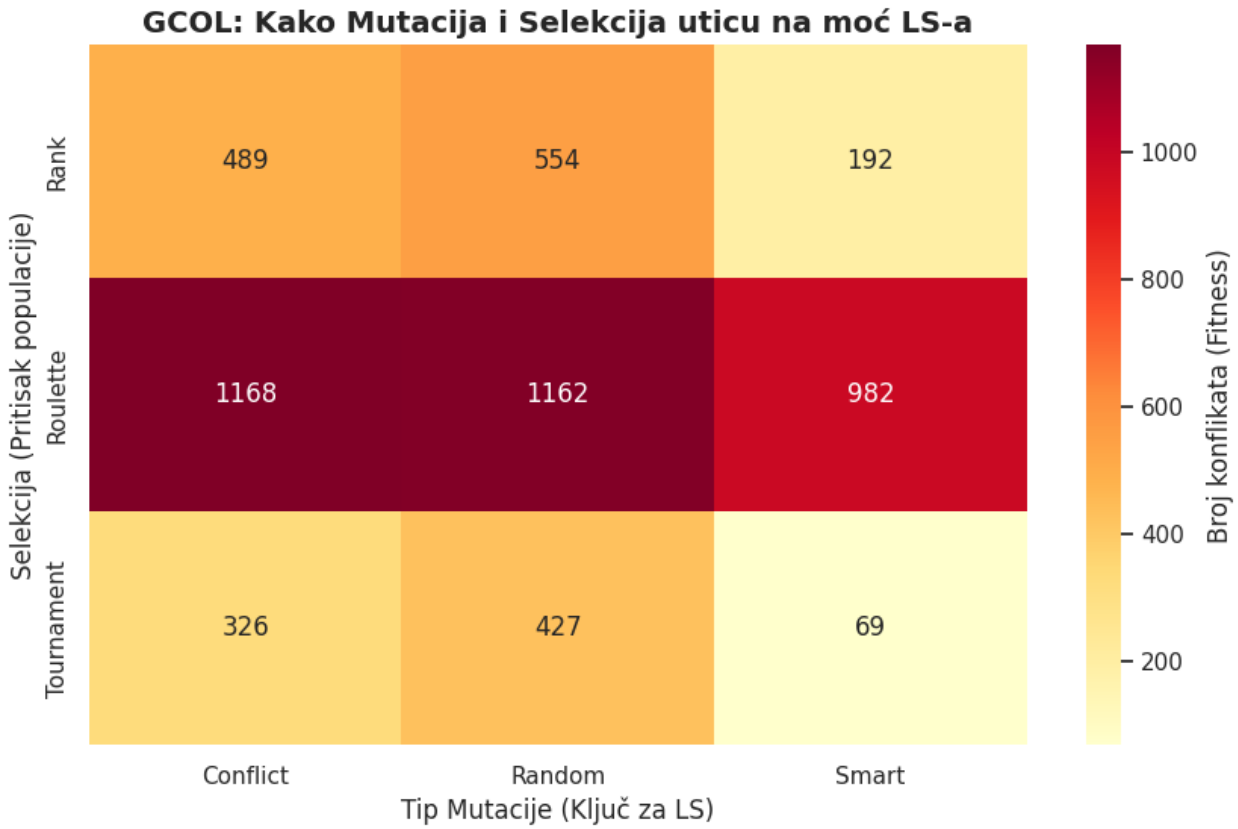
Na osnovu toplotne mape, uočava se jasna prednost upotebe **Tournament selekcije u kombinaciji sa Conflict mutacijom**. Dok su nasumični mehanizmi, poput Roulette selekcije, pokazali tendenciju ka stagnaciji u lokalnim optimumima, Tournament selekcija održava stabilan napredak populacije. Učinak je dodatno maksimizovan Conflict mutacijom, koja za razliku od ostalih operatora direktno adresira ivice sa konfliktom, čime se postiže brža i stabilnija optimizacija regularnih grafova visokog stepena.

RandomGNP - random generisan graf sa nx.gnp_random_graph(n,p) - gde su n broj čvorova i p verovatnoća za nastanak ivice.



Testiranjem algoritma na nasumičnim grafovima visoke gustine ispitali smo njegovu otpornost na ekstremna opterećenja, što se najbolje uočava na primeru grafa sa **85 čvorova i 2129 ivica,** uz maksimalni stepen čvora **Δ=62**. Uporedni prikaz na toplotnoj mapi omogućava nam da pri takvoj gustini precizno identifikujemo limite određenih genetskih operatora. Ovi podaci su ključni za razumevanje mogućnosti algoritma pri rešavanju najsloženijih problema bojenja ivica u realnim sistemima.

Finalni deo istraživanja sproveden je na zahtevnoj **GCOL instanci** koja broji **100 čvorova i 2487 ivica,** uz maksimalni stepen čvora **Δ=61**. Ovako visoka vrednost delte, u kombinaciji sa ogromnim brojem ivica, stvara ekstremno zasićen graf u kojem je pronalaženje legalnog bojenja bez konflikata značajno teže nego u prethodnim slučajevima.



Na osnovu prikupljenih podataka, identifikovano je da kombinacija **Tournament selekcije i Smart mutacije** ostvaruje najmanju prosečnu vrednost fitness funkcije.

Zaključak

Na osnovu svih sprovedenih testova, može se zaključiti da performanse hibridnog algoritma presudno zavise od nivoa selektivnog pritiska i preciznosti mutacije. **Roulette selekcija** se ubedljivo pokazala kao najlošiji izbor u svim scenarijima, dok se **Tournament selekcija** izdvaja kao apsolutni favorit, pružajući neophodan intenzitet pretrage kroz sve klase grafova.

Kada je reč o rekombinaciji materijala, operatori ukrštanja generalno pokazuju slične performanse, mada **Two-point crossover** ostvaruje malu prednost u efikasnosti, čineći ga najboljim izborom. U domenu mutacije, **Smart i Conflict** operatori su se pokazali kao najpouzdaniji jer, za razliku od nasumičnih pristupa, direktno ciljaju konflikte i time generišu kvalitetne ulaze za dalju optimizaciju.

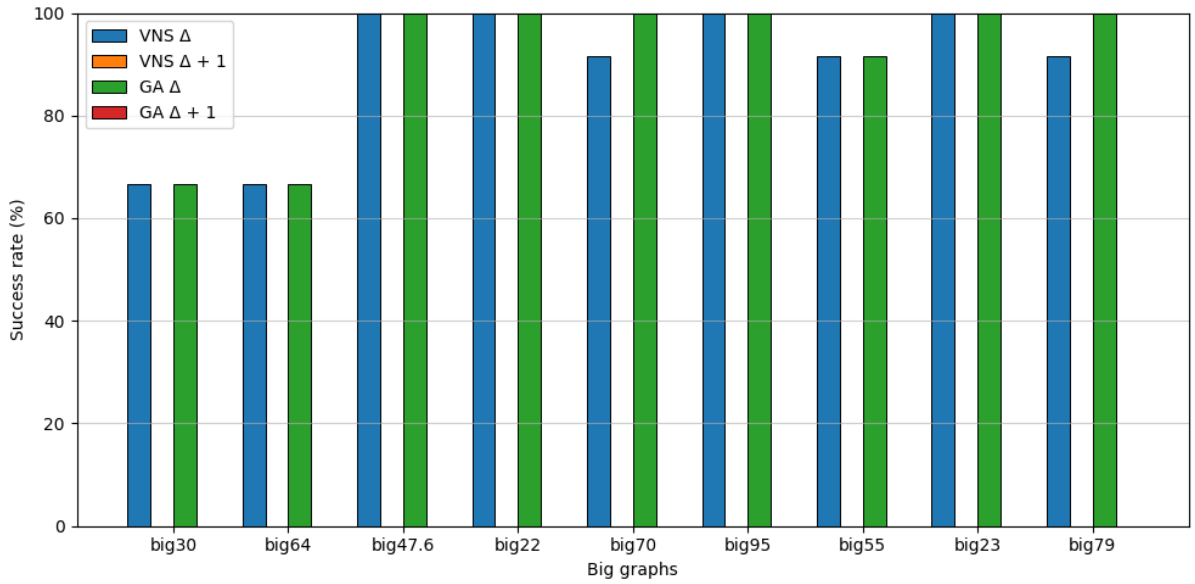
Posebno kritičnu ulogu igra **lokalna pretraga (LS)**. Rezultati na teškim instancama (GCOL, random GNP 85) potvrđuju da bi bez localSearchHARD komponente bilo praktično nemoguće eliminisati konflikte u grafovima visoke gustine. Dok genetski operatori služe za globalnu navigaciju kroz prostor rešenja, lokalna pretraga je ta koja daje rezultate i omogućava algoritmu da se nosi sa ekstremnim ograničenjima.

Testiranje je izvršeno na laptopu sa sledećim specifikacijama:

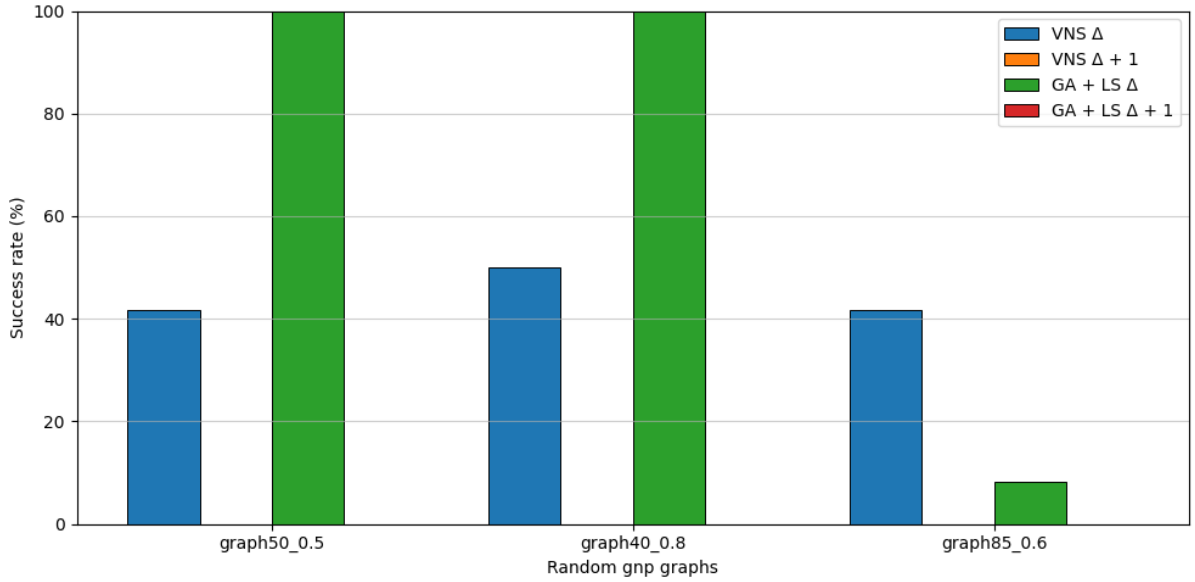
- **Procesor:** Intel® Core™ i5-1035G1 CPU @ 1.00GHz × 8
- **Operativni sistem:** Ubuntu 22.04.5 LTS
- **Programski jezik i alati:** Python, CPLEX 20.1 rešavač

Poredjenje VNS i GA + LS

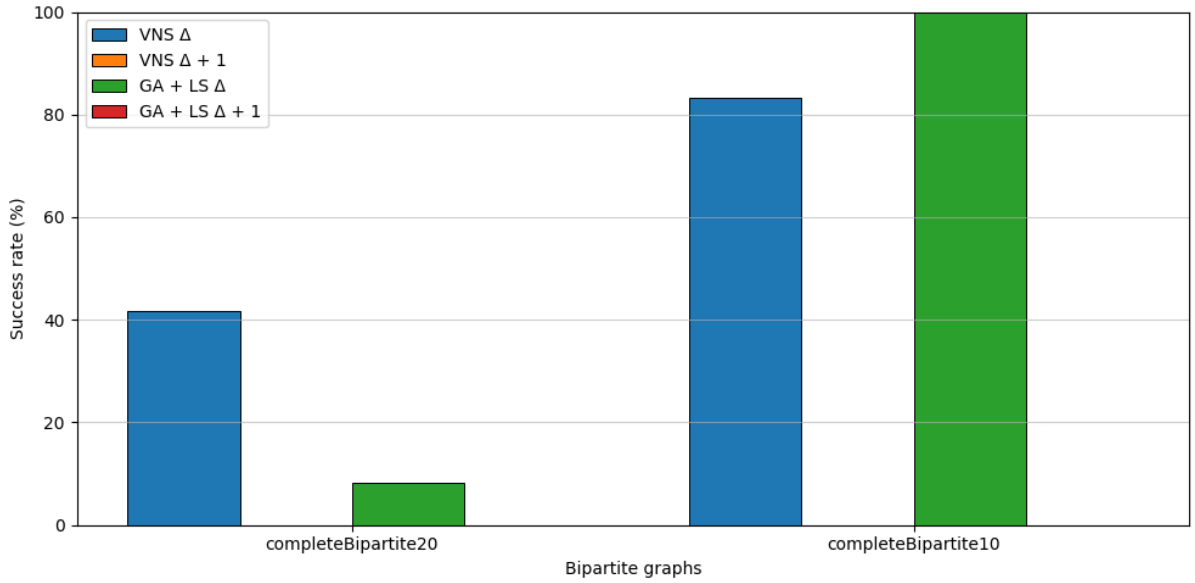
Svaka implementacija je imala vise mogucih kombinacija rešavanja problema. Ovde računamo kolika je uspešnost kombinacija za svaku od dve metode. Grafovi su podeljeni u podgrupe kao i pre. Ukoliko postoji barem ijedno Δ bojenje, procenat Δ+1 bojenja neće biti računato.



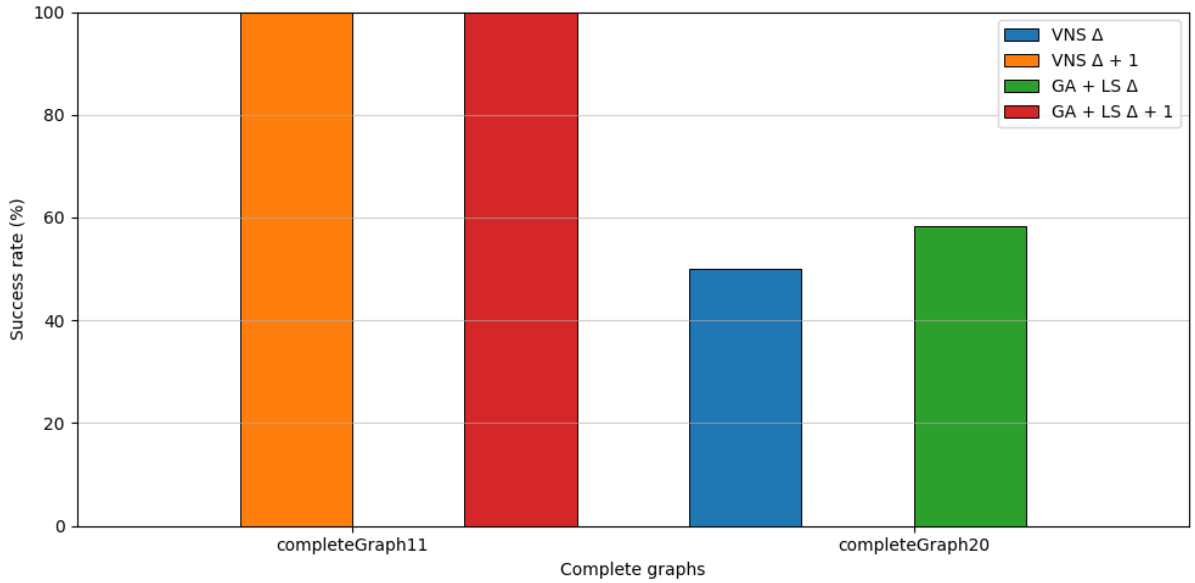
Skoro za svaki graf procenat upešnih Δ bojenja je isti. Ipak, za nijansu je bolji GA + LS



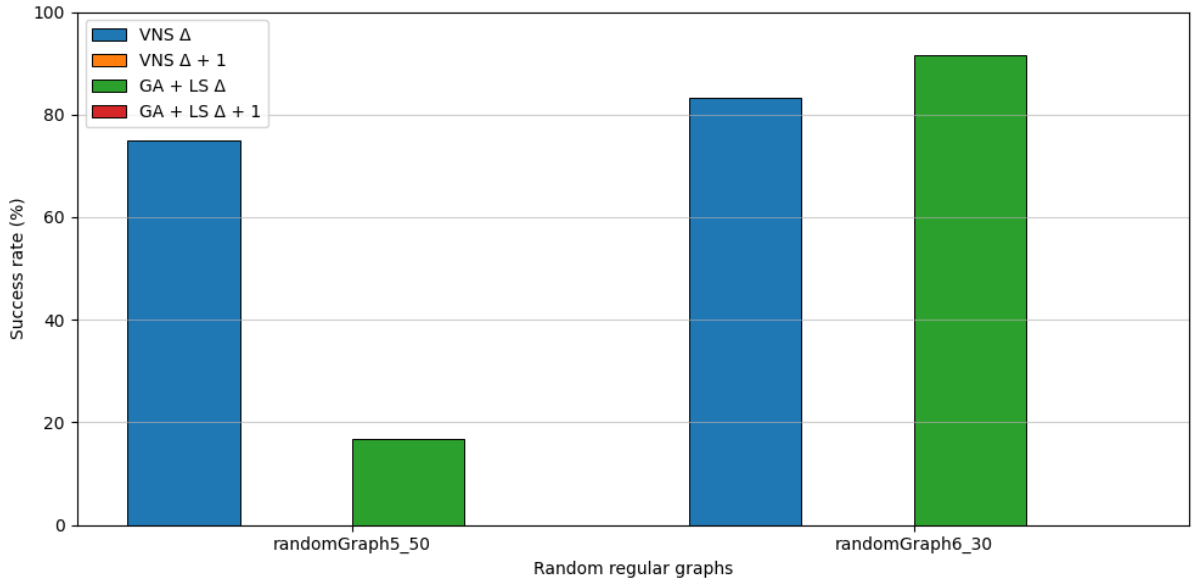
Za prva dva grafa, GA + LS ima 100% učinak. Mada, kod komplikovanijeg grafa se VNS bolje pokazao.



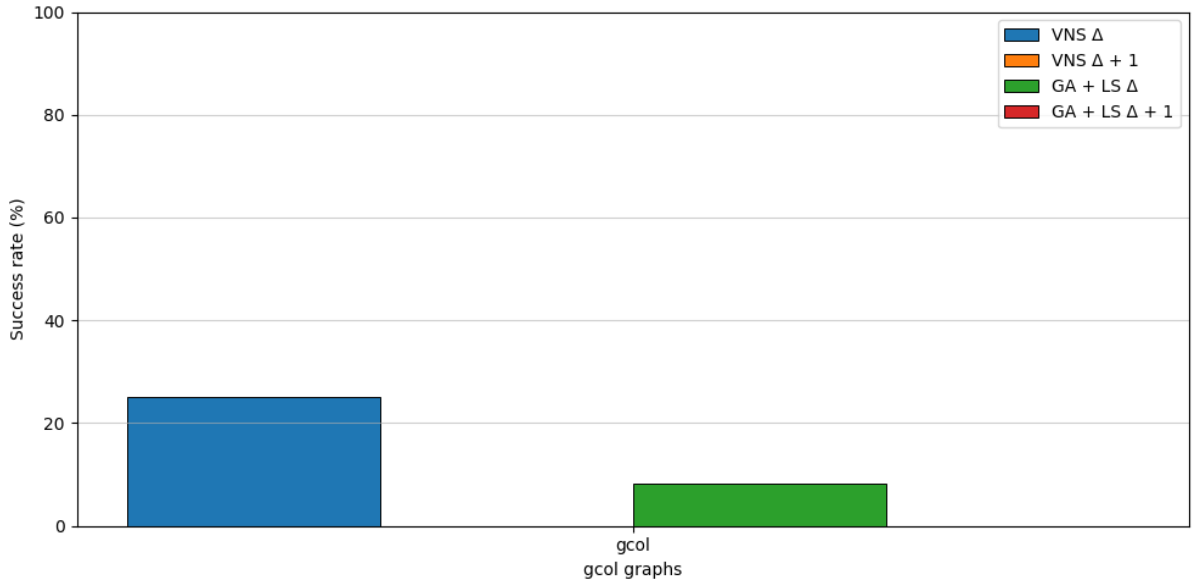
Takodje, kod komplikovanijeg grafa se VNS bolje pokazao.



Complete graph 11 ne može da se oboji sa Δ boja, pa sa Δ + 1 sve kombinacije obe optimizacije su uspele da nadju rešenje. Kod complete20 ipak GA+LS odnosi pobedu.



Jako velika razlika u procentu uspešnosti kod randomRegular5_50.



Obe optimizacije imaju mali procenat uspešnosti. Ipak, VNS pobedjuje.

Zaključak:

U većini grupa grafova GA+LS se pokazao bolje. Medjutim, mora se primetiti da pri kompleksnijim grafovima više kombinacija VNS uspeva da nadje Δ bojenje.

Reference

1. https://www.csc.kth.se/~viggo/wwwcompendium/node18.html
2. https://www.lirmm.fr/~bessy/GraphesStructM1/DM3/Papers/LevenGalil.pdf
3. https://www.jstage.jst.go.jp/article/iis/1/1/1_1_19/_pdf/-char/en
4. https://www.labri.fr/perso/mbonamy/917U/3-Edge-Colouring.pdf
5. https://www.gerad.ca/~alainh/VNSEJOR.pdf
6. https://perso.limos.fr/~palafour/PAPERS/PDF/Garey-Johnson79.pdf
7. https://www.sciencedirect.com/science/article/pii/S1018363913000135
8. https://ceur-ws.org/Vol-841/submission_10.pdf

Instance

1. https://mat.tepper.cmu.edu/COLOR/instances.html
2. https://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/