Nalaženje maksimalne nezavisne sekvence

Aleksandra Radosavljević

Matematički Fakultet, Univerzitet u Beogradu

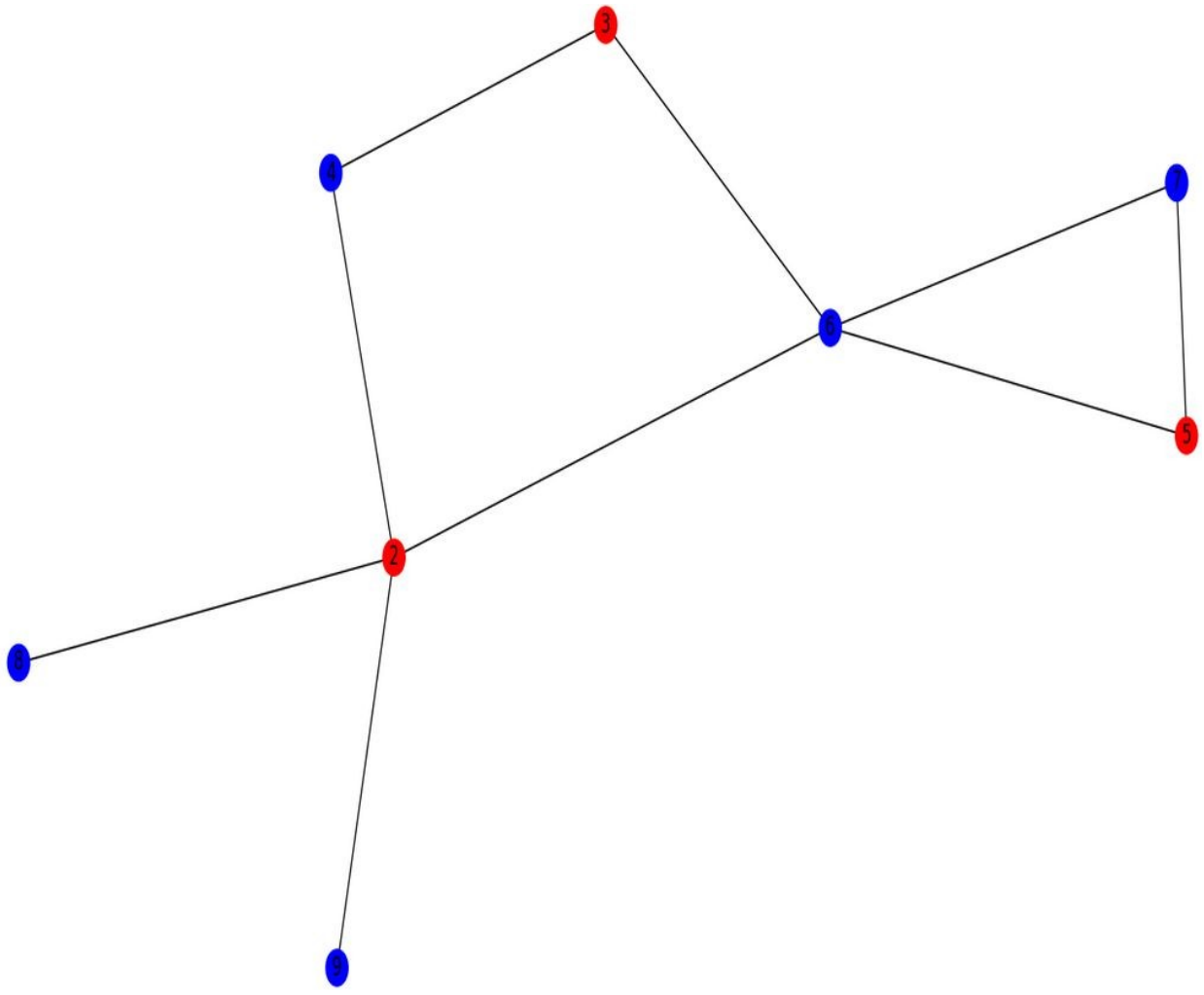
Februar 2022

1 Uvod

Nezavisna sekvenca za graf G = (V, E) je niz čvorova *v1, v2, ..., v_n* takvih da za svaki čvor v_i postoji neki susedan čvor u koji nije susedan ni sa jednim čvorom v_j, $j \leq i$. [1]

Najduži takav niz naziva se **maksimalna nezavisna sekvenca.**

Nalaženje maksimalne nezavisne sekvence dakle podrazumeva trazenje nezavisnog niza čvorova u grafu sa najvećom kardinalnošću.



Graf u kome je jedno od rešenja niz [3, 2, 5].

2 Predložena rešenja

Svi pristupi su implementirani u programskom jeziku Python 3 uz biblioteku Networkx za kreiranje i internu reprezentaciju grafa, i Matplotlib za prikazivanje grafova, kao i grafkona za testiranje i poredjenje.

2.1 Gruba sila – Naivni pristup

Da bismo pronašli rešenje grubom silom, moramo da proverimo svaku permutaciju čvorova. Za graf G = (V, E) će složenost ovakvog pristupa biti O(|V|!), pa ovakav pristup nije pogodan u realnim okolnostima. Ipak ovakav pristup sigurno daje optimalan rezultat.

Broj grana i ivica	Prosečno vreme
8	2s
9	26s
10	277s (4 min)
11	3234s (55min)

2.2 Metaheuristički pristup – Simulirano kaljenje

Pristup iterativne optimizacije rešenja simuliranim kaljenjem se čini primenljivo za rešavanje ovakvog problema. Da bi se čvor razmatrao kao član rešenja potreban mu je barem jedan susedan čvor koji nije susedan sa dosadašnjim članovima sekvence, stoga čvorovi manjeg stepena imaju male šanse da udju kao kasniji članovi sekvence, pa je razmatrano sortiranje u rastućem poretku pre početka optimizacije, medjutim, nije bilo prevelikog odstupanja od običnog pristupa za

male grafove, ali s povećavanjem veličine grafa vidi se napredak. Bolji rezultati su ipak primećeni pri kaljenu koje brže konvergira što govori da možda i lokalna pretraga daje dovoljno dobro rešenje.

2.3 Metaheuristički pristup – Genetski algoritam

Genetski algoritam znatno sporije izvršava izračunavanje zato što, za razliku od simuliranog kaljenja, vrednost se izračunava u svakoj iteraciji za svaku jedinku u svojoj populaciji. Vreme se može ubrzati smanjivanjem populacije kao i smanjivanjem iteracija. Za potrebe testiranja, korišćeno je jednopoziciono ukrštanje, verovatnoća mutacije 2% kao i turnirska selekcija veličine 5. U prvi mah, veličina populacije je bila 100 jediniki, medjutim u cilju poboljšanja vremena izračunavanja smanjena je populacija na 50.

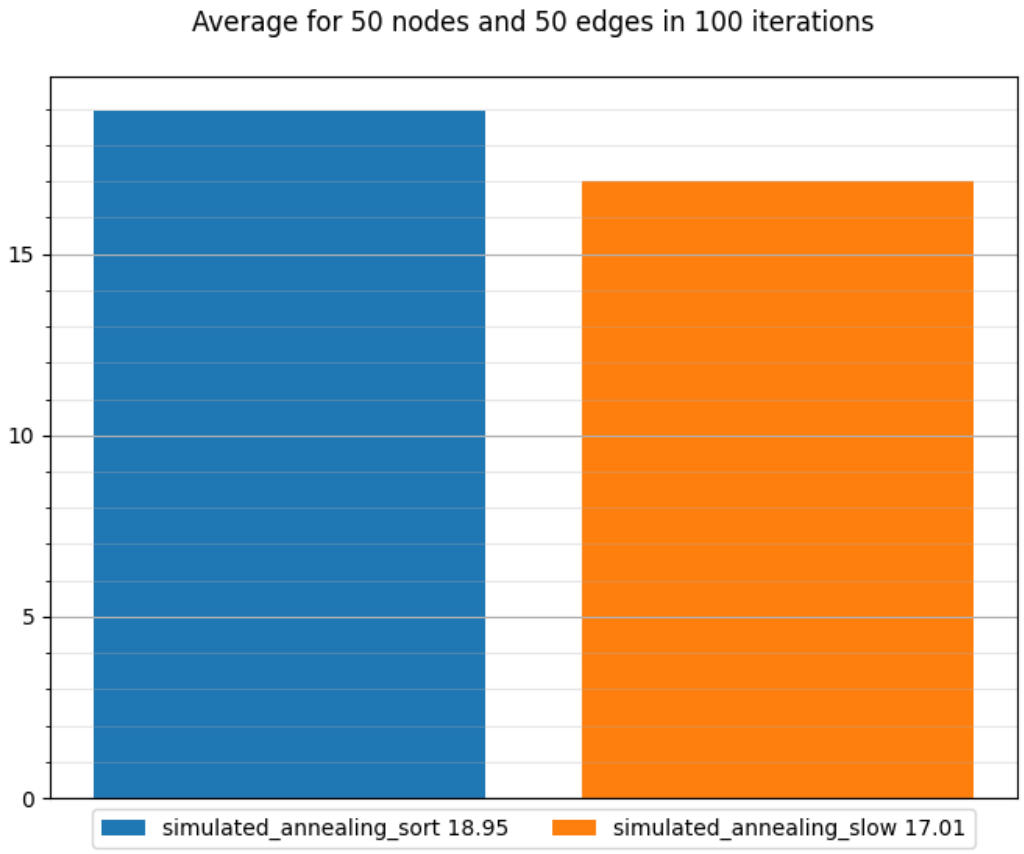
3 Testiranje rešenja

Sva testiranja su radjena na računaru sa Windows 10 (64-bit) operativnim sistemom sa procesorom AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx 2.00 GHz.

3.1 Analiza pojedinačnih rešenja

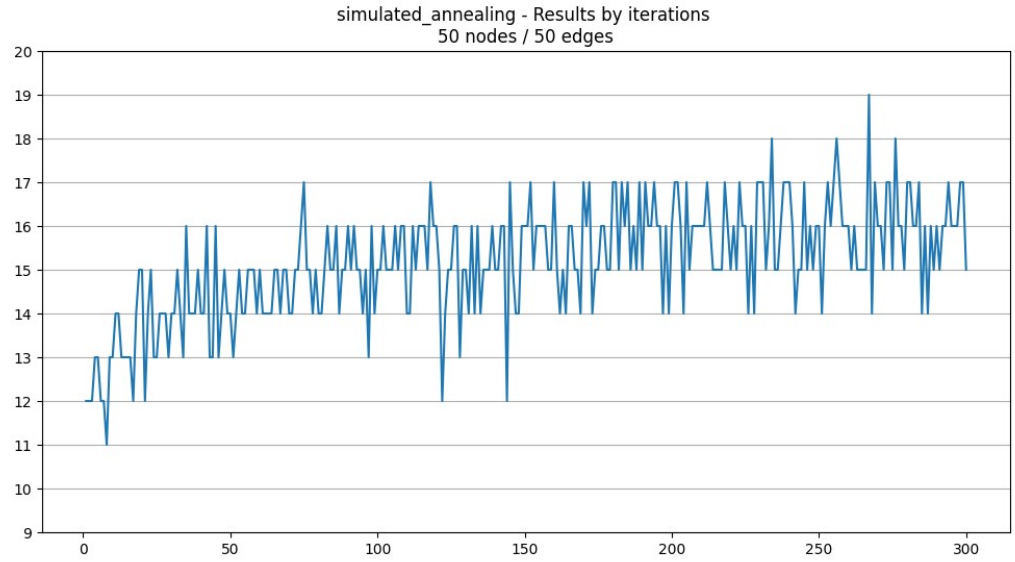
3.1.1 Simulirano kaljenje

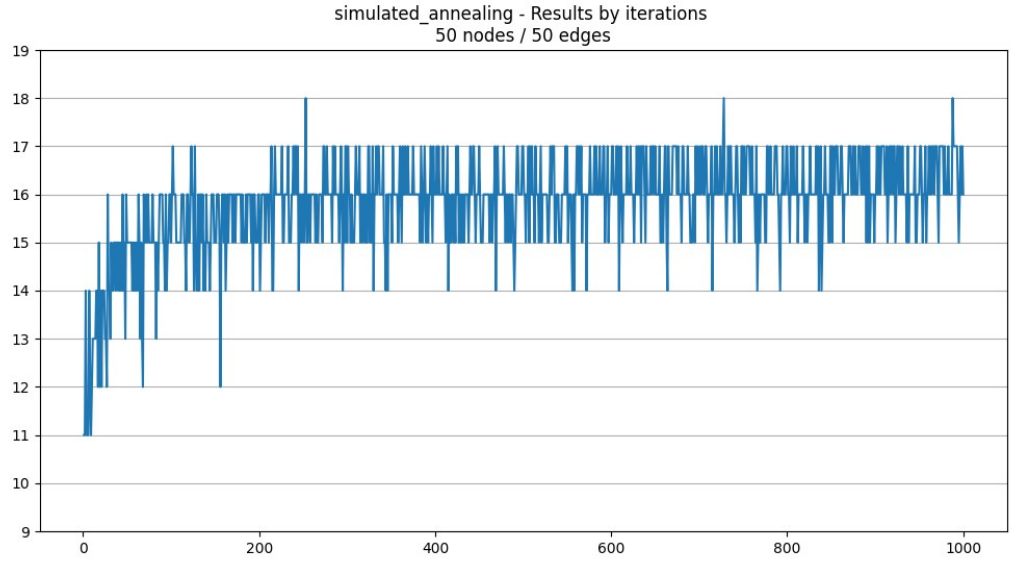
Različite brzine: Testiranjem je zakljuceno da se simulirano kaljenje bolje ponasa kada konvergira brze. Prikazane su varijante kada su verovatnoce prihvatanja resenja $1/i$ i $1/\sqrt{i}$ gde je i broj iteracije u tom trenutku. Na 100 iteracija, prvi pristup je dao bolje resenje u čak 87 slučaja, drugi samo 3,a 10 puta su dali isti rezultat.



Kaljenje sa funkcijama prihvatanja i lošijeg rešenja: a) *1/i* b) *1/Vi*

Broj iteracija utiče na rešenje: Možemo primetiti da broj iteracija veoma utiče na kvalitet rešenja. Vidimo da sa povećanjem broja iteracija dobijamo bolja rešenja iako je mogće da povremeno nazadujemo sa kvalitetom. Vidimo na oba grafika da je maksimum dostignut na oko 250 iteracija.





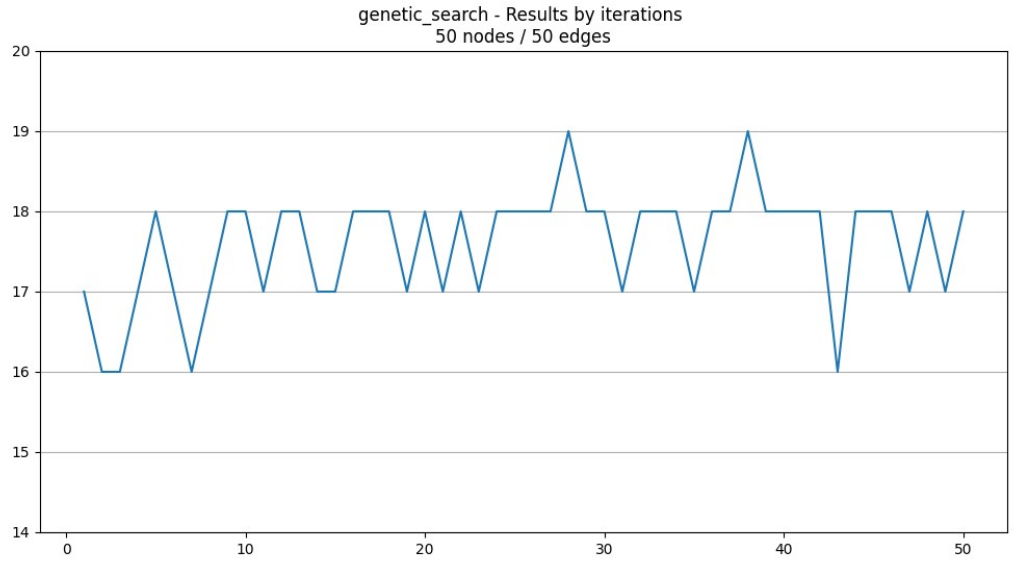
Sortiranje čvorova pre početka: Sortiranje čvorova pre početka dovelo je do znatnog poboljšanja u rezultatu. Pri poredjenju u 1000 iteraciija sa 20 čvorova i 20

grana 8 puta rezultat bio bolji u algoritmu obicnog kaljenja,35 puta u algoritmu kaljenja sa sortitanjem ,a 57 puta je rezultat bio jednak.

3.1.2 Genetski algoritam

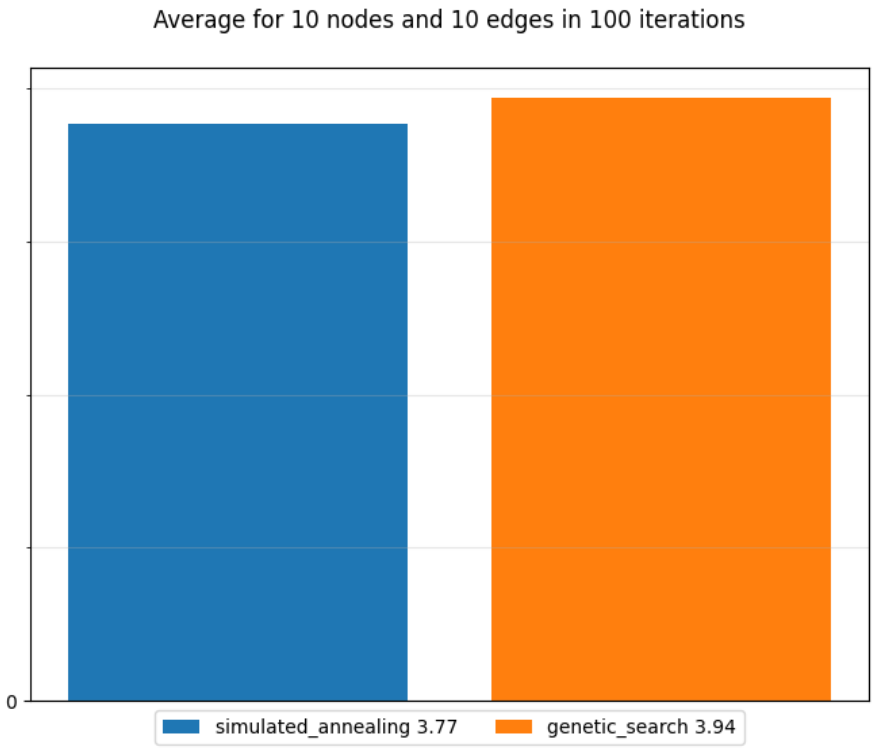
Elitizam: Pri testiranju genetskog algoritma korišćen je pristup elitizma od 30% populacije, medjutim preimećeno je da se najbolji rezultat mnogo puta ponavlja u kasnijim generacijama ,a kako je potrebno naći globalni maksimum, a ne lokalni ,elitizam je izbačen iz razmatranja.

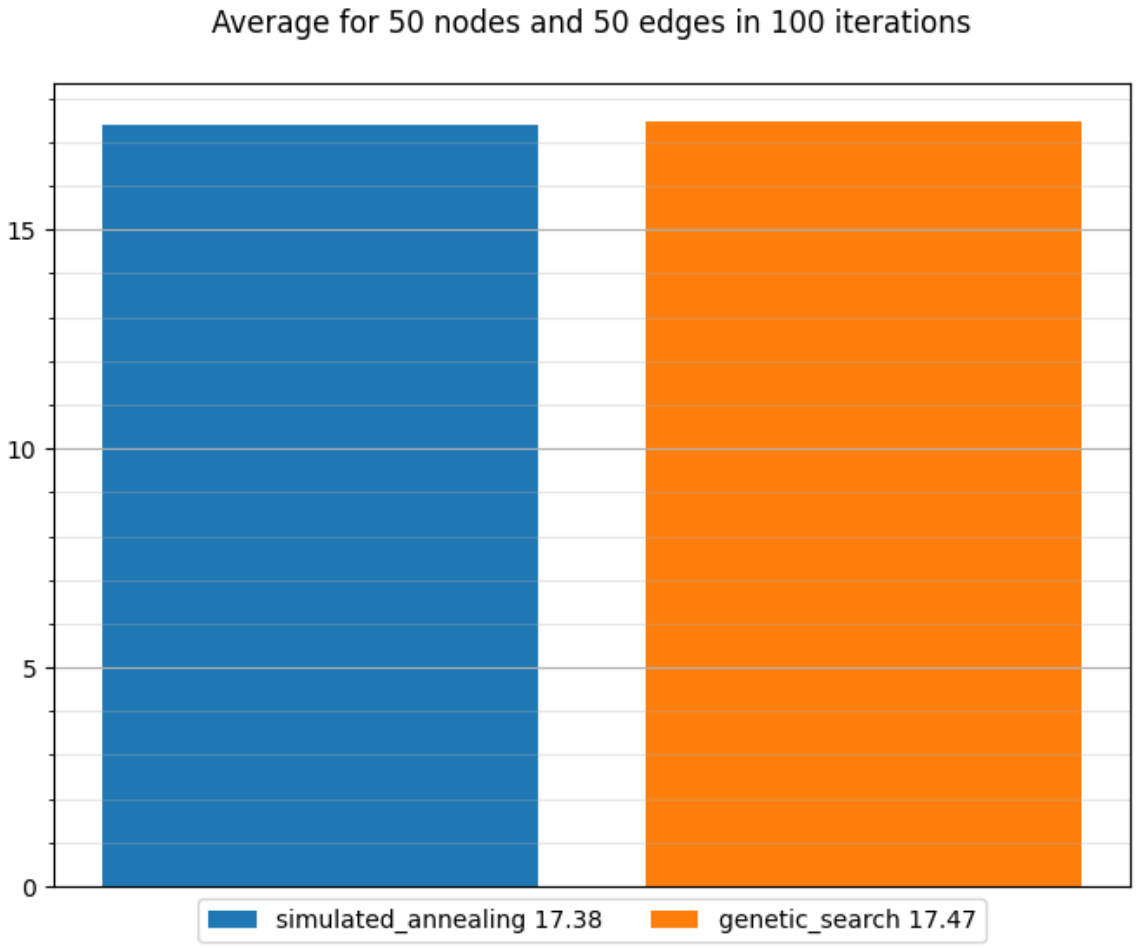
Broj iteracija: Kao i kod kaljenja, genetski algoritam takodje daje bolje rezultate sa porastom iteracija,ipak taj broj nije toliko bitan zato što genetski računa rezultate cele jedne generacije u jednoj iteraciji dok kod kaljenja imamo samo jedan rezultat po iteraciji.

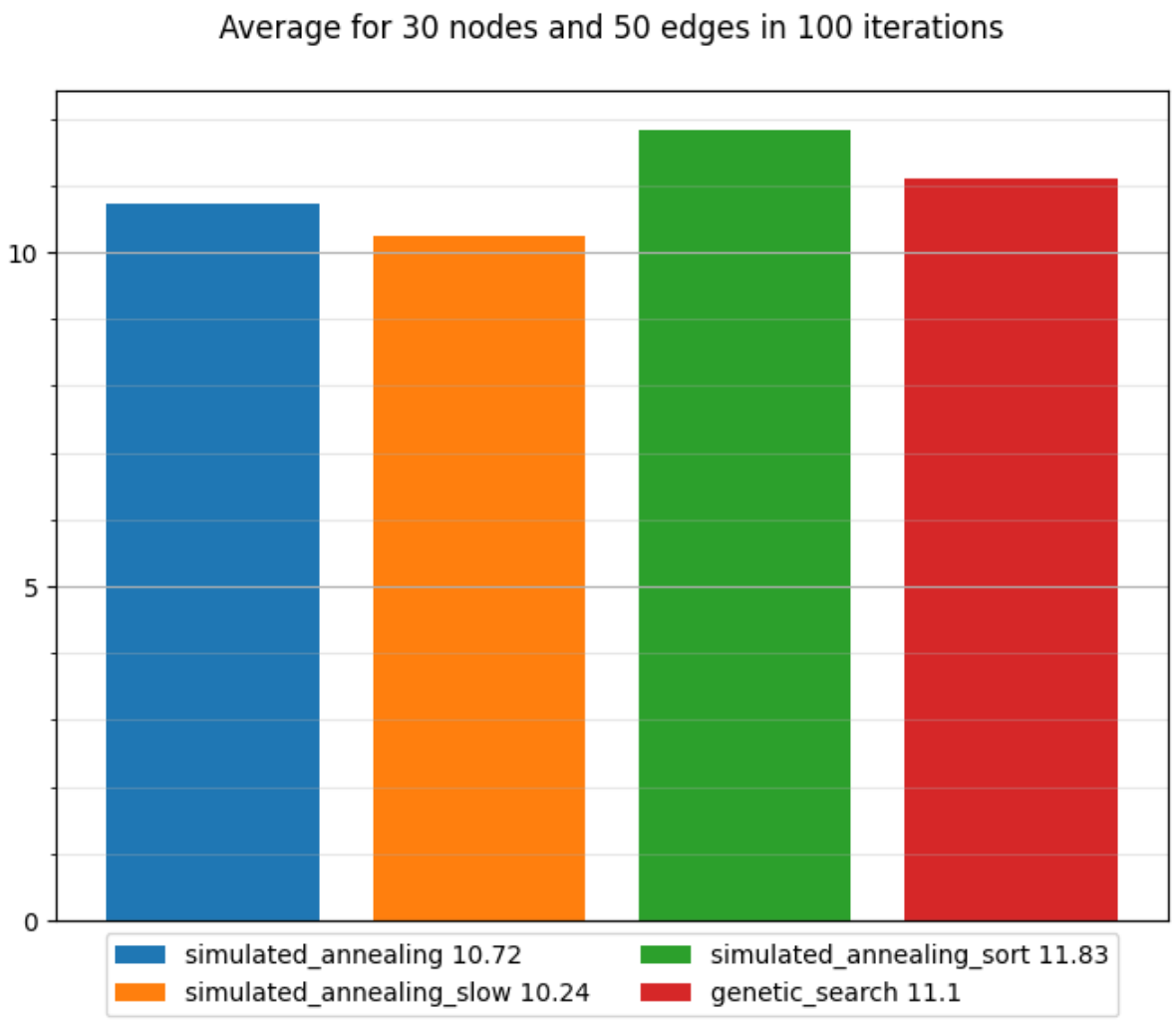


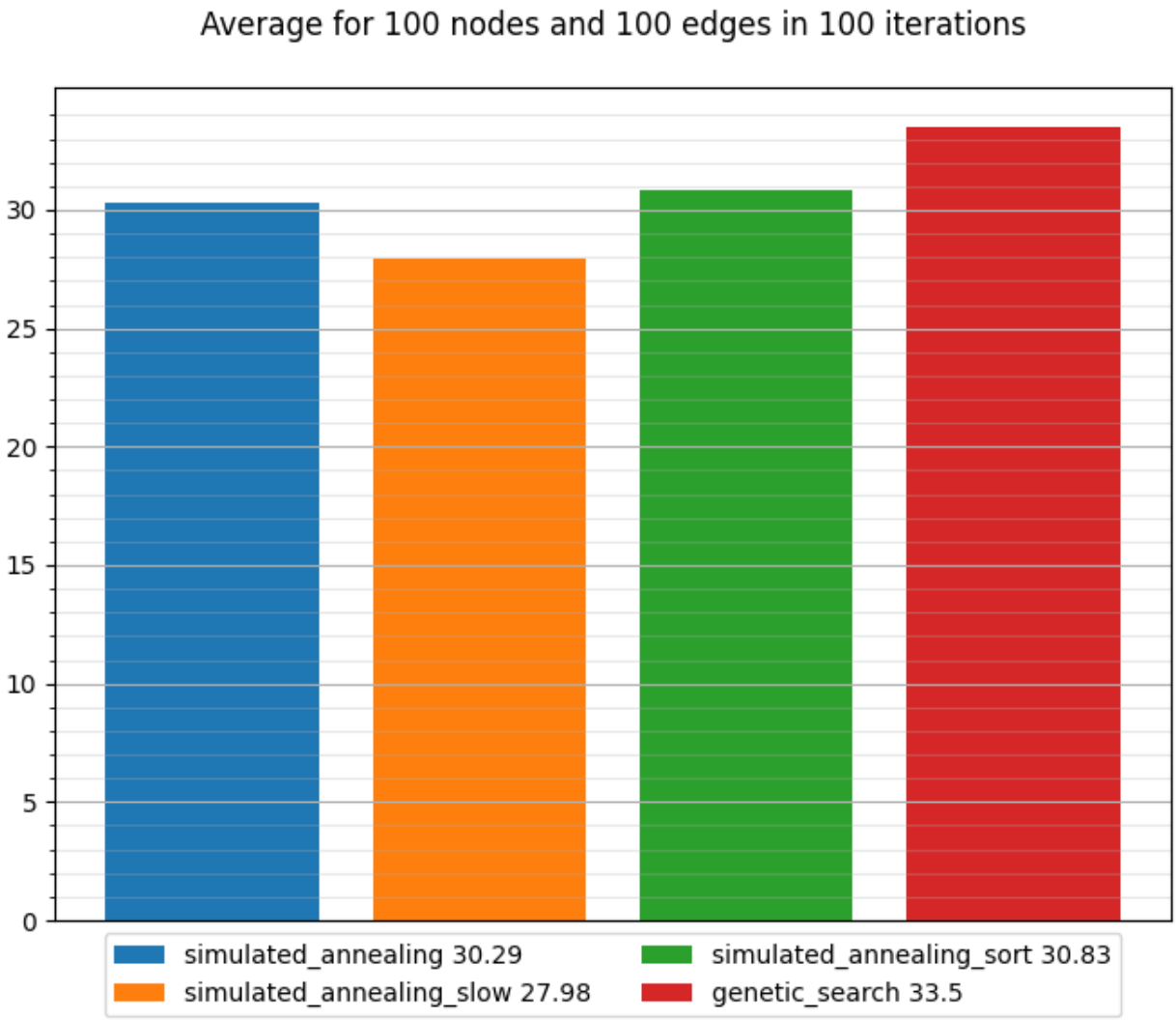
Ovde vidimo da su najbolja rešenja dobijena onda kada je genetskom algoritmu prosledjeno izmedju 25 I 40 iteracija koje predstavljaju broj generacija

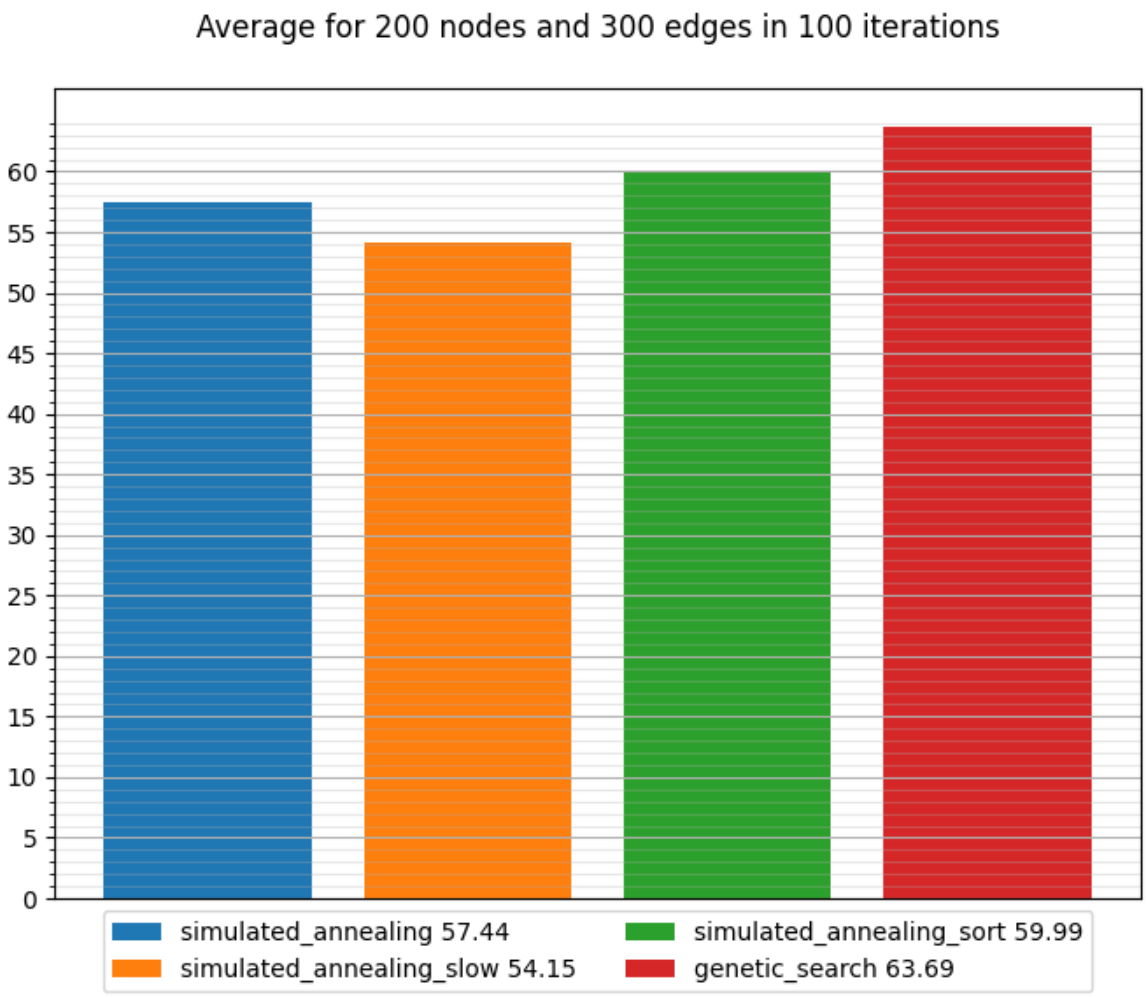
3.2. Razna poređenja




Average for 50 nodes and 50 edges in 100 iterations
15
10
5
0
simulated_annealing 17.38
genetic_search 17.47


Average for 30 nodes and 50 edges in 100 iterations
10
5
0
simulated_annealing 10.72
simulated_annealing_slow 10.24
simulated_annealing_sort 11.83
genetic_search 11.1


Average for 100 nodes and 100 edges in 100 iterations
0
5
10
15
20
25
30
simulated_annealing 30.29
simulated_annealing_slow 27.98
simulated_annealing_sort 30.83
genetic_search 33.5



Primećujemo da genetski algoritam daje najbolja rešenja, dok obično kaljenje i kaljenje sa sortiranjem daju slične rezultate. Znatna razlika se primećuje na grafovima sa velikim brojem čvorova i ivica.

Zaključak

Na osnovu predhodnih rezultata možemo zaključiti da je genetski algoritam definitivno najbolji izbor u rešavanju problema maksimalne nezavisne sekvence. Sama činjenica da se cela generacija ocenjuje kroz zadati broj iteracija daje veliku raznolikost rešenja gde je potrebno da se samo jedna jedinka istakne.

Takodje prednost genetskog algoritma u odnosu na kaljenje ima u činjenici da jedinke prolaze kroz fazu ukrštanja kao i fazu mitacije,dok u kaljenju možemo reći da imaju samo sigurnu fazu mutacije.

Preporuka je pri rešavanju problema grafa sa malim brojem čvorova i ivica koristiti kaljenje pošto je razlika zanemarljiva, a kaljenje je algoritam manje složenosti, dok se sa porastom čvorova i ivica genetski algoritam nameće kao optimalno rešenje.

Potencijalno poboljšanje bi moglo biti otkriveno promenom izbora okoline kod kaljenja,da ne bude nasumicno, već da se okolina bira nekom heuristikom. To ostaje neistraženo za sada i ostavlja se kao sugestija čitaocu.

Literatura

Refrences

[1] P. Crescenzi and V. Kann. A compendium of NP optimization problems,

maximum independent sequence.1999.

https://cs.pwr.edu.pl/zielinski/lectures/om/compendium.pdf