

Rešavanje problema maksimalne nezavisne sekvence

Stefan Mirić
Matematički Fakultet, Univerzitet u Beogradu

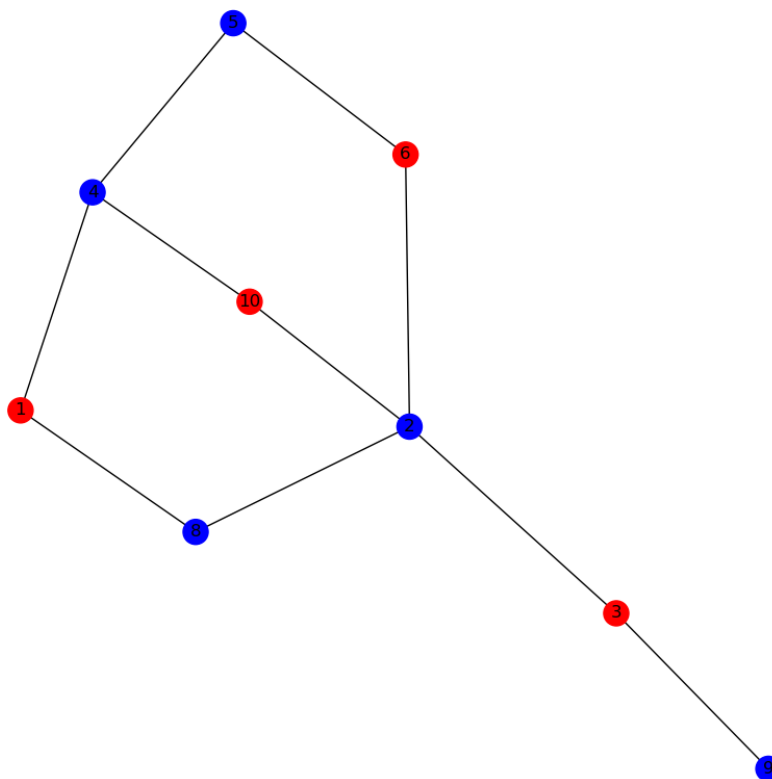
Januar 2020

1 Uvod

Nezavisna sekvenca za graf $G=(V,E)$ je niz čvorova v_1, v_2, \dots, v_m takvih da za svaki v_{i+1} postoji neki susedan čvor u koji nije susedan ni sa jednim čvorom v_j , $j \leq i$.

Takav niz, koji sadrži najveći broj elemenata naziva se **maksimalna nezavisna sekvenca**.

Problem maksimalne nezavisne sekvence podrazumeva traženje nezavisnog niza čvorova u grafu sa najvećom kardinalnošću, odnosno traženje nezavisne sekvence koja je najveća.



Primer grafa u kome je jedno od rešenja niz $[10, 3, 6, 1]$

Primetimo da su rešenja ovog problema nasledna: naime ako je v_1, v_2, \dots, v_m nezavisna sekvenca, onda će i svaka podsekvenca $v_{a_1}, v_{a_2}, \dots, v_{a_x}$ biti nezavisna.

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 grafikona za testiranje i poredjenje

2.1 Gruba sila - Naivni pristup

Da bismo pronasli rešenje grubom silom, moraćemo da proverimo svaku mogucu permutaciju čvorova. Za graf $G=(V,E)$, jasno je da ce slozenost 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	3s
9	35s
10	7min
11	oko 3 sata

Tabela vremena izvršavanja algoritma grube sile

2.2 Metaheuristički pristup - Simulirano kaljenje

Pristup iterativne optimizacije rešenja simuliranim kaljenjem izgleda primenljivo pri rešavanju ovakvog problema. Kako je čvoru potreban barem jedan susedan čvor koji nije susedan sa dosadašnjim članovima sekvence da bi se razmatrao kao član rešenja, čvorovi manjeg stepena imaju male šanse da uđu kao kasniji članovi sekvence, pa je razmatrano sortiranje u rastućem poretku pre početka optimizacije, međutim, nije bilo prevelikog odstupanja od običnog pristupa. Bolji rezultati su ipak primećeni pri kaljenju koje brže konvergira što govori da je možda i lokalna pretraga daje dovoljno dobro rešenje. Razmatrano je i drugačije biranje okoline, ali ne i testirano.

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 prilike testiranja, korišćeno je ukrštanje prvog reda, verovatnoća mutacije 1% kao i turnirska selekcija. U prvi mah, veličina populacije je bila 100 jedinki, međutim u cilju poboljšanja vremena izračunavanja smanjena je populacija na 50.

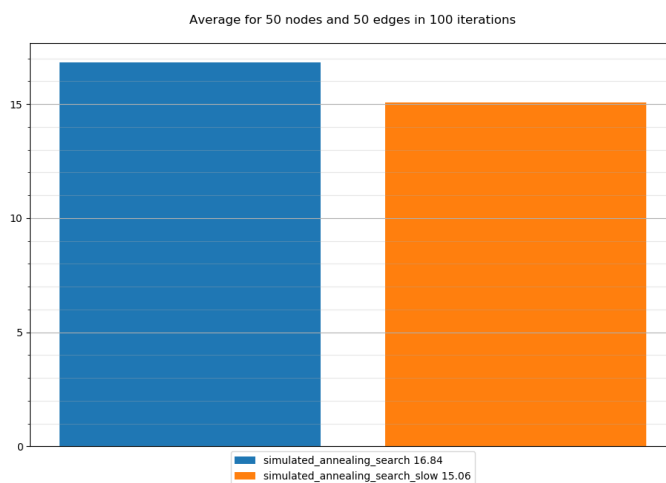
3 Testiranje rešenja

Sva testiranja rađena su na računaru sa Windows 7 operativnim sistemom i sa Intel-ovim dvojezgarnim procesorom G2030 sa 3.0 GHz

3.1 Analiza pojedinačnih rešenja

3.1.1 Simulirano kaljenje

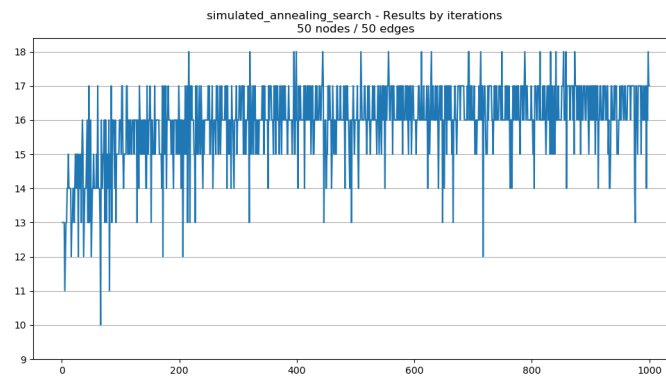
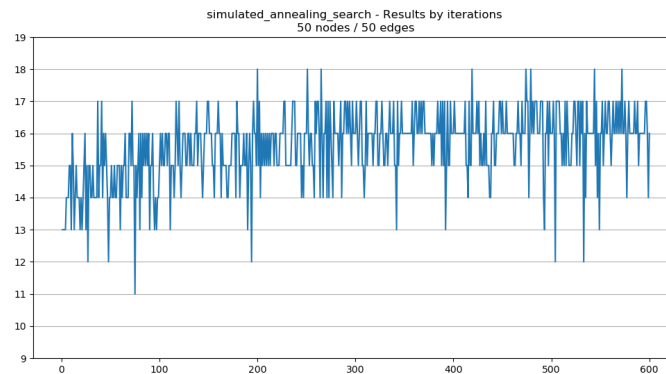
Različite brzine konvergiranja: Testiranjem je zaključeno da se simulirano kaljenje bolje ponaša kada konvergira brže. Prikazane su varijante kada su verovatnoće prihvatanja rešenja $\frac{1}{i}$ i $\frac{1}{\sqrt{i}}$ gde je i broj iteracije u tom trenutku. Na 100 iteracija, prvi pristup je dao bolje rešenje u čak 80 slučajeva,



Kaljenje sa funkcijama prihvatanja lošijeg rešenja: a) $\frac{1}{i}$ b) $\frac{1}{\sqrt{i}}$

drugi u samo 4, a 16 puta su dali isti rezultat.

Broj iteracija utiče na rešenje: Mozemo primetiti da broj iteracija veoma utice na kvalitet rešenja. Vidimo da sa povećanjem iteracija dobijamo bolja rešenja iako je moguće da povremeno nazadujemo sa kvalitetom. Vidimo na oba grafika kako dostižemo maksimum na oko 200 iteracija



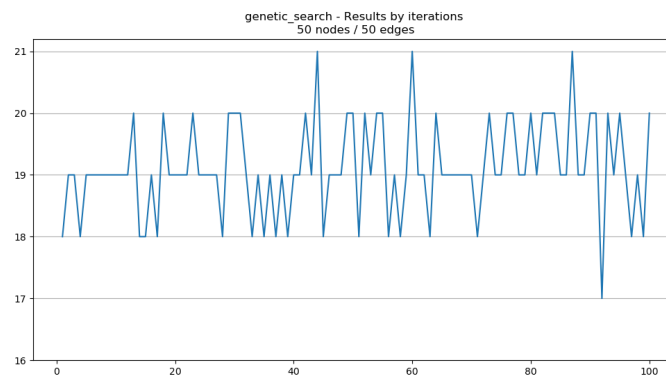
Kaljenje do 600 i 1000 iteracija

Sortiranje čvorova pre početka: Nije bilo znakova poboljšanja u odnosu na obično kaljenje. 38 puta bolji rezultat je dalo normalno kaljenje, 31 put kaljenje sa sortiranjem i isto toliko puta je rezultat bio jednak.

3.1.2 Genetski algoritam

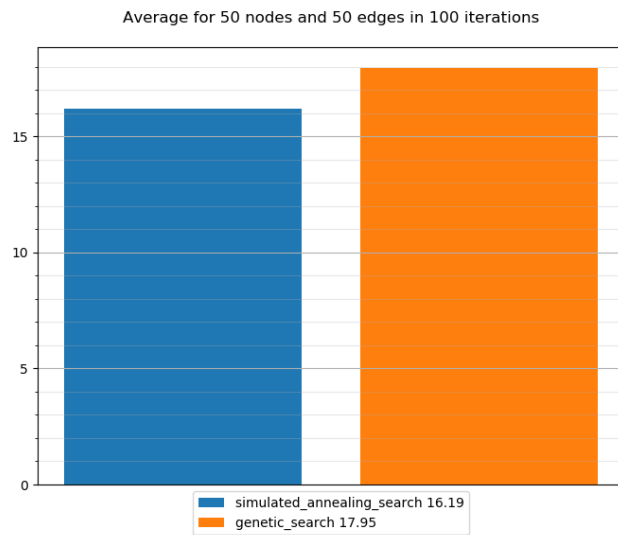
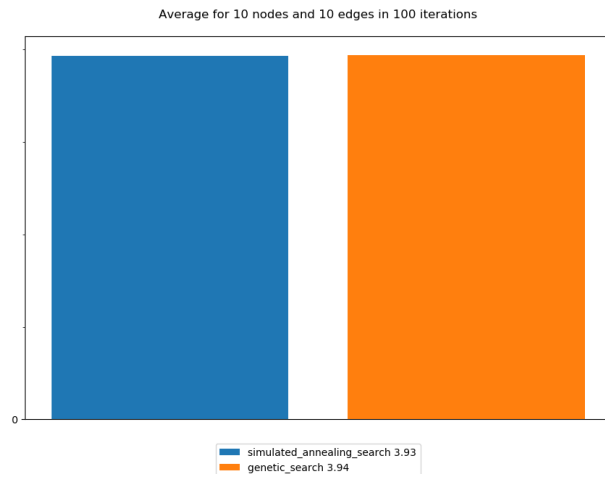
Elitizam: Pri testiranju genetskog algoritma korišćen je pristup elitizma od 30% populacije, međutim primećeno je da se najbolji rezultat mnogo puta ponavlja u kasnijim generacijama, a kako je potrebno naći globalni maksimum, a ne lokalni, elitizam je izbačen iz razmatranja.

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

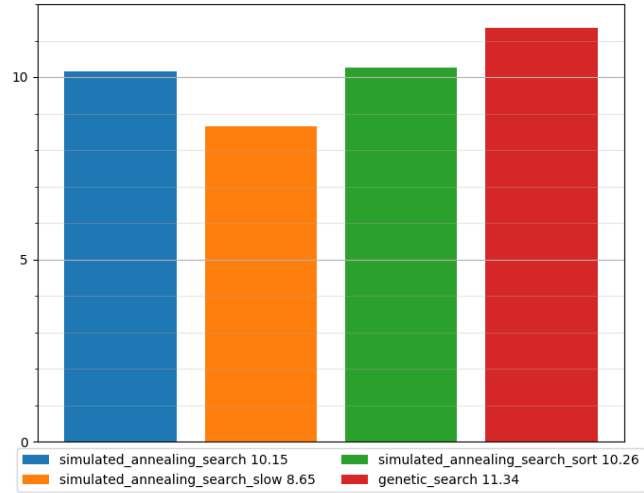


Vidimo da mnogo manje variraju rešenja nego kod kaljenja

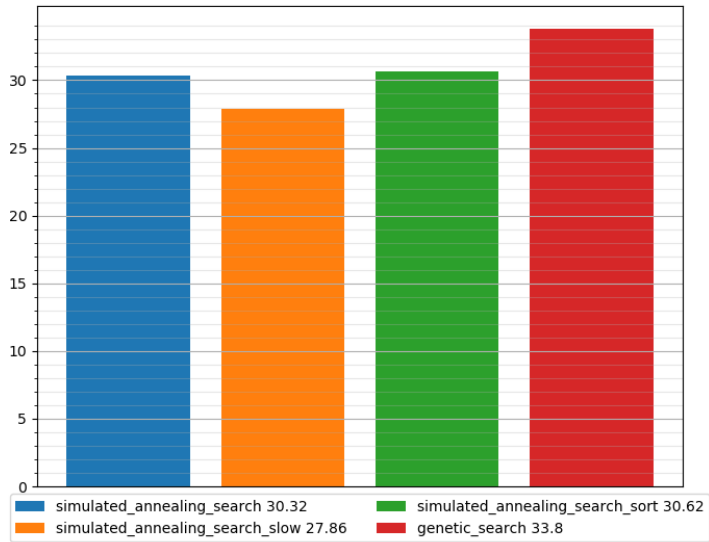
3.2 Razna poređenja



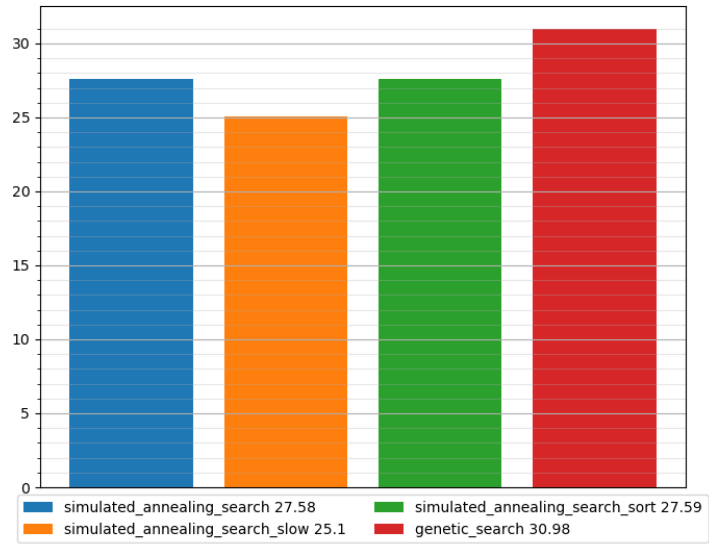
Average for 30 nodes and 50 edges in 100 iterations



Average for 100 nodes and 100 edges in 100 iterations



Average for 100 nodes and 250 edges in 100 iterations



Average for 200 nodes and 300 edges in 100 iterations

