

Istraživanje i optimizacija minimalnog nezavisnog dominantnog skupa u grafovima

Sara Kalinić 387/2021

September 27, 2024

1 Uvod

Minimalni nezavisni dominantni skup čvorova je važan problem u teoriji grafova i računarstvu. Ovaj problem je NP-težak, što znači da ne postoji poznat polinomni algoritam koji može rešiti sve instance ovog problema. U ovom radu istražujemo različite heurističke tehnike za optimizaciju rešenja, uključujući varijantna okruženja pretraživanja (VNS), genetske algoritme i optimizaciju mravima.

2 Teorijski Okvir

2.1 Definicija

Nezavisni dominantni skup čvorova u grafu $G = (V, E)$ je podskup čvorova $D \subseteq V$ tako da su svi čvorovi u V ili u D ili su susedi nekog čvora iz D .

3 Implementacija Rešenja

3.1 Grubom Silom

Grubom silom je metoda koja se koristi za pronalaženje minimalnog nezavisnog dominantnog skupa čvorova u grafu pretraživanjem svih mogućih podskupova čvorova. U ovoj sekciji objašnjava se implementacija ove metode koristeći Python i biblioteku NetworkX.

3.1.1 Opis Koda

Za implementaciju grube sile razvijen je Python kod koji se oslanja na kombinacije čvorova u grafu. Ključni delovi koda uključuju:

- **Definisanje funkcija:**

- `is_independent_set(graph, subset)`: Ova funkcija proverava da li je dati `subset` nezavisni skup. Prolazi kroz sve moguće kombinacije čvorova u `subset` i proverava da li postoji ivica izmeu bilo koja dva čvora. Ako postoji ivica, skup nije nezavistan.

- `is_dominating_set(graph, subset)`: Ova funkcija proverava da li `subset` dominira ceo graf. Ona prikuplja sve čvorove iz `subset` i njihove susede, i proverava da li je ukupan broj čvorova u `dominating_nodes` jednak broju čvorova u grafu.
- `brute_force_mids(graph)`: Ova funkcija implementira glavnu logiku grube sile. Ona generiše sve moguće podskupove čvorova koristeći funkciju `itertools.combinations` i proverava svaki podskup da li ispunjava uslove za nezavisni i dominantni skup. Ako se nađe manji skup koji zadovoljava uslove, on se čuva kao najbolji rezultat.

• Vizualizacija grafa:

- Funkcija `draw_graph(graph, min_set)` koristi biblioteku Matplotlib za vizualizaciju grafa. Čvorovi koji čine minimalni nezavisni dominantni skup se prikazuju u crvenoj boji, dok su ostali čvorovi u svetloplavoj boji. Ova vizualizacija pomaže u boljem razumevanju strukture grafa i pozicije odabranih čvorova.

3.1.2 Rezultati

Na grafu koji se učitava iz datoteke `tests/test_30_0.3.in`, poziv funkcije `brute_force_mids(G)` vraća minimalni nezavisni dominantni skup. Rezultat se štampa u konzoli i vizualizuje pomoću funkcije `draw_graph`.

U sledećoj tabeli prikazani su rezultati dobijeni primenom grube sile na grafovima sa različitim brojem čvorova i verovatnoćom veze.

Test Name	Broj čvorova	Broj iteracija	Vreme (s)	Rezultat
test_10_0.3.in	2	1023	0.0009999	Minimalni nezavisni dominantni skup
test_10_0.5.in	3	1023	0.0009994	Minimalni nezavisni dominantni skup
test_10_0.7.in	2	1023	0.0	Minimalni nezavisni dominantni skup
test_15_0.3.in	4	32767	0.025	Minimalni nezavisni dominantni skup
test_15_0.5.in	2	32767	0.025	Minimalni nezavisni dominantni skup
test_20_0.3.in	5	1048575	1.509	Minimalni nezavisni dominantni skup
test_20_0.5.in	3	1048575	0.684	Minimalni nezavisni dominantni skup
test_25_0.3.in	4	33554431	29.311	Minimalni nezavisni dominantni skup
test_25_0.5.in	3	33554431	22.975	Minimalni nezavisni dominantni skup
test_25_0.7.in	2	33554431	22.999	Minimalni nezavisni dominantni skup
test_30_0.3.in	0	0	0.0	Graf prevelik za grubu silu
test_30_0.5.in	0	0	0.0	Graf prevelik za grubu silu
test_35_0.3.in	0	0	0.0	Graf prevelik za grubu silu
test_35_0.5.in	0	0	0.0	Graf prevelik za grubu silu
test_40_0.3.in	0	0	0.0	Graf prevelik za grubu silu
test_40_0.5.in	0	0	0.0	Graf prevelik za grubu silu
test_40_0.7.in	0	0	0.0	Graf prevelik za grubu silu

Table 1: Rezultati testiranja za minimalni nezavisni dominantni skup

Ova metoda, iako jednostavna i intuitivna, može biti vrlo spora za velike grafove zbog eksponencijalne kompleksnosti. Ipak, koristi se kao osnova za poređenje s drugim heurističkim metodama u daljoj analizi.

3.2 Variable Neighborhood Search (VNS)

Variable Neighborhood Search (VNS) je heuristička metoda optimizacije koja istražuje više susednih prostora rešenja kako bi pronašla bolje rešenje. U ovoj sekciji objašnjavamo implementaciju VNS algoritma za pronalaženje minimalnog nezavisnog dominantnog skupa čvorova u grafu.

3.2.1 Opis Koda

Implementacija VNS algoritma obuhvata sledeće ključne funkcije:

- **Inicijalizacija:**

- Funkcija `initialize(graph)` generiše nasumično rešenje, gde svaki čvor u grafu ima 30% verovatnoće da bude uključen u rešenje. Ovo se postiže korišćenjem `random.random()` funkcije.

- **Fitness funkcija:**

- Funkcija `calc_fitness(solution, graph)` izračunava fitness vrednost datog rešenja. Ako je rešenje nezavisni i dominantni skup, vraća se recipročna vrednost broja čvorova u skupu. U suprotnom, vraća se negativna beskonačnost.

- **Lokalna pretraga:**

- Funkcija `local_search_invert_best_improvement(solution, value, graph)` primenjuje lokalnu pretragu da poboljša trenutno rešenje. Ona invertuje svaki bit u rešenju i proverava da li dobijeno novo rešenje ima bolju fitness vrednost. Ako se pronađe bolje rešenje, ono se čuva i proces se nastavlja.

- **Shaking:**

- Funkcija `shaking(solution, k)` kreira novo rešenje tako što nasumično menja `k` bitova u trenutnom rešenju. Ova funkcija pomaže u istraživanju novih podružja rešenja.

- **VNS glavni algoritam:**

- Funkcija `vns(graph, vns_params)` implementira VNS algoritam. Za svaku vrednost `k` (od `k_min` do `k_max`), generiše se novo rešenje koristeći funkciju `shaking`, koje se zatim poboljšava lokalnom pretragom. Ako novo rešenje ima bolju fitness vrednost, ono postaje novo trenutno rešenje.

3.2.2 Rezultati

Funkcija `vns_main(graph)` poziva glavni VNS algoritam sa unapred definisanim parametrima. Na kraju, najbolji rezultat se prikazuje u konzoli, a graf se vizualizuje koristeći funkciju `draw_graph`.

Najbolji fitnes: (rezultati)

Minimalni Nezavisni Dominirajući Skup: (rezultati)

Ova metoda pokazuje značajnu efikasnost u pronalaženju rešenja za problem minimalnog nezavisnog dominantnog skupa, uz potencijal za dodatna unapređenja kroz fino podešavanje parametara.

Test Name	Broj čvorova	Broj iteracija	Vreme (s)	Rezultat
test_10_0.3.in	2	10000	2.539	Minimalni nezavisni dominantni skup
test_10_0.5.in	3	10000	1.745	Minimalni nezavisni dominantni skup
test_10_0.7.in	2	10000	1.606	Minimalni nezavisni dominantni skup
test_15_0.3.in	4	10000	2.702	Minimalni nezavisni dominantni skup
test_15_0.5.in	2	10000	2.445	Minimalni nezavisni dominantni skup
test_20_0.3.in	9	10000	3.995	Nema validnog rešenja
test_20_0.5.in	3	10000	3.517	Minimalni nezavisni dominantni skup
test_25_0.3.in	4	10000	5.409	Minimalni nezavisni dominantni skup
test_25_0.5.in	3	10000	4.212	Minimalni nezavisni dominantni skup
test_25_0.7.in	2	10000	3.533	Minimalni nezavisni dominantni skup
test_30_0.3.in	4	10000	4.693	Minimalni nezavisni dominantni skup
test_30_0.5.in	3	10000	4.725	Minimalni nezavisni dominantni skup
test_35_0.3.in	6	10000	6.524	Minimalni nezavisni dominantni skup
test_35_0.5.in	11	10000	6.022	Nema validnog rešenja
test_40_0.3.in	13	10000	8.592	Nema validnog rešenja
test_40_0.5.in	14	10000	7.846	Nema validnog rešenja
test_40_0.7.in	8	10000	6.439	Nema validnog rešenja

Table 2: Rezultati testiranja za minimalni nezavisni dominantni skup koristeći VNS

3.3 Genetski Algoritam (GA)

Genetski algoritam (GA) je optimizacijska tehnika inspirisana prirodnom selekcijom, koja koristi procese evolucije za rešavanje složenih problema. U ovoj sekciji detaljno opisujemo implementaciju GA za pronalaženje minimalnog nezavisnog dominantnog skupa čvorova u grafu.

3.3.1 Opis Koda

Implementacija genetskog algoritma se sastoji iz nekoliko ključnih klasa i funkcija:

- **Klasa Individual:**

- Klasa `Individual` predstavlja jedno rešenje problema. Svaki `Individual` sadrži kod (niz binarnih vrednosti) koji predstavlja uključene čvorove, i izračunava svoju fitness vrednost kroz metodu `calc_fitness`.
- Funkcije `is_independent_set` i `is_dominating_set` proveravaju da li je skup čvorova nezavistan ili dominantan.

- **Selekcija:**

- Funkcija `selection(population, tournament_size)` koristi turnirsku selekciju da odabere roditelje za kreiranje novog naraštaja. Izvodi se nasumični uzorak iz populacije, a najbolji pojedinac se bira kao roditelj.

- **Kros-over:**

- Funkcija `crossover(parent1, parent2, child1, child2)` kombinuje kodove dva roditelja u dva nova potomka na osnovu nasumične tačke preseka.

- **Mutacija:**

- Funkcija `mutation(individual, mutation_prob)` nasumično menja kodove pojedinaca kako bi se uvela genetska raznolikost, čime se sprečava stagnacija algoritma.

- **Genetski algoritam:**

- Funkcija `ga(population_size, num_generations, tournament_size, elitism_size, mutation_prob, graph)` implementira glavni proces genetskog algoritma. U svakoj generaciji, populacija se sortira, elitizam se primenjuje, roditelji se biraju, kros-over i mutacija se primenjuju kako bi se stvorili novi pojedinci.

3.3.2 Rezultati

Funkcija `ga_main(graph)` pokreće genetski algoritam sa unapred definisanim parametrima. Na kraju, najbolji rezultat se prikazuje u konzoli, a graf se vizualizuje koristeći funkciju `draw_graph`.

```
--- %s seconds ---
```

```
Najbolji fitnes: (rezultati)
```

Test Name	Broj čvorova	Broj iteracija	Vreme (s)	Rezultat
test_10_0.3.in	2	10000	2.313	Minimalni nezavisni dominantni skup
test_10_0.5.in	3	10000	2.083	Minimalni nezavisni dominantni skup
test_10_0.7.in	2	10000	2.198	Minimalni nezavisni dominantni skup
test_15_0.3.in	4	10000	2.298	Minimalni nezavisni dominantni skup
test_15_0.5.in	2	10000	2.535	Minimalni nezavisni dominantni skup
test_20_0.3.in	6	10000	2.346	Minimalni nezavisni dominantni skup
test_20_0.5.in	3	10000	2.129	Minimalni nezavisni dominantni skup
test_25_0.3.in	5	10000	2.280	Minimalni nezavisni dominantni skup
test_25_0.5.in	4	10000	2.236	Minimalni nezavisni dominantni skup
test_25_0.7.in	2	10000	2.259	Minimalni nezavisni dominantni skup
test_30_0.3.in	8	10000	2.433	Nema validnog rešenja
test_30_0.5.in	18	10000	3.023	Nema validnog rešenja
test_35_0.3.in	16	10000	2.340	Nema validnog rešenja
test_35_0.5.in	11	10000	2.712	Nema validnog rešenja
test_40_0.3.in	14	10000	2.484	Nema validnog rešenja
test_40_0.5.in	17	10000	3.202	Nema validnog rešenja
test_40_0.7.in	15	10000	2.297	Nema validnog rešenja

Table 3: Rezultati testiranja za minimalni nezavisni dominantni skup koristeći genetski algoritam

Ova metoda pokazuje efikasnost u pronalaženju minimalnog nezavisnog dominantnog skupa, sa sposobnošću da istraži široku oblast rešenja kroz evolucijske procese.

3.4 Optimizacija pomoću Metode Mravlje Kolonije (ACO)

Metoda Mravlje Kolonije (ACO) je bioinspirisana optimizacijska tehnika koja koristi ponašanje mrava u potrazi za hranom kako bi rešila složene probleme. U ovoj sekciji opisujemo implementaciju ACO za pronalaženje minimalnog nezavisnog dominantnog skupa čvorova u grafu.

3.4.1 Opis Koda

Implementacija ACO sadrži nekoliko ključnih funkcija:

- **is_independent_set:**
 - Ova funkcija proverava da li je dati skup čvorova nezavistan. Ako postoji ivica izmeu bilo koja dva čvora u skupu, funkcija vraća **False**.
- **is_dominating_set:**
 - Ova funkcija proverava da li je dati skup čvorova dominantan. Svi čvorovi u grafu moraju biti ili u skupu ili susedni čvorovima u skupu.
- **fitness:**
 - Funkcija **fitness** izračunava fitness vrednost rešenja. Ako je rešenje nezavisno i dominantno, fitness se izračunava kao inverz vrednosti broja čvorova u skupu, u suprotnom se vraća negativna vrednost.
- **aco_minimum_independent_dominating_set:**
 - Ova funkcija implementira ACO. U svakom iteraciji, svaki mrav gradi rešenje na osnovu feromona i preostalih čvorova. Vrednost feromona se ažurira u zavisnosti od fitness vrednosti rešenja.

3.4.2 Rezultati

Funkcija `aco_minimum_independent_dominating_set(graph, num_ants, num_iterations, alpha, beta, evaporation_rate)` pokreće ACO sa unapred definisanim parametrima. Na kraju, najbolji rezultat se prikazuje u konzoli, a graf se vizualizuje koristeći funkciju `draw_graph`.

Best Independent Dominating Set: (rezultati)
Fitness (1 / length of the set): (rezultati)

Ova metoda se pokazala kao efikasna u pronalaženju minimalnog nezavisnog dominantnog skupa, omogućavajući istraživanje rešenja kroz iteracije i feromone.

Test Name	Broj čvorova	Broj iteracija	Vreme (s)	Rezultat
test_10_0.3.in	3	1000	2.013	Minimalni nezavisni dominantni skup
test_10_0.5.in	3	1000	2.216	Minimalni nezavisni dominantni skup
test_10_0.7.in	2	1000	1.447	Minimalni nezavisni dominantni skup
test_15_0.3.in	5	1000	3.121	Minimalni nezavisni dominantni skup
test_15_0.5.in	3	1000	2.644	Minimalni nezavisni dominantni skup
test_20_0.3.in	6	1000	4.151	Minimalni nezavisni dominantni skup
test_20_0.5.in	3	1000	2.240	Minimalni nezavisni dominantni skup
test_25_0.3.in	5	1000	3.710	Minimalni nezavisni dominantni skup
test_25_0.5.in	3	1000	3.349	Minimalni nezavisni dominantni skup
test_25_0.7.in	2	1000	2.802	Minimalni nezavisni dominantni skup
test_30_0.3.in	5	1000	4.258	Minimalni nezavisni dominantni skup
test_30_0.5.in	4	1000	3.378	Minimalni nezavisni dominantni skup
test_35_0.3.in	4	1000	3.874	Minimalni nezavisni dominantni skup
test_35_0.5.in	3	1000	3.178	Minimalni nezavisni dominantni skup
test_40_0.3.in	7	1000	5.141	Minimalni nezavisni dominantni skup
test_40_0.5.in	4	1000	4.388	Minimalni nezavisni dominantni skup
test_40_0.7.in	3	1000	3.550	Minimalni nezavisni dominantni skup

Table 4: Rezultati testiranja za minimalni nezavisni dominantni skup koristeći Ant Colony Optimization

4 Uporedna Analiza

U ovoj sekciji uporeujemo rezultate svih implementacija. Tabela 5 prikazuje performanse različitih pristupa.

Table 5: Uporedna tabela rezultata algoritama

Algoritam	Prosečno vreme (sekundi)	Stopa uspešnosti (%)
VNS	3.64	70.59
GA	2.27	58.82
BRUT	7.65	55.56
ACO	3.12	100.00

5 Zaključak

U ovom istraživanju, analizirali smo različite heurističke metode za pronalaženje minimalnog nezavisnog dominantnog skupa čvorova u grafu, uključujući VNS, GA, BRUT i ACO. Na osnovu dobijenih rezultata možemo izvući nekoliko značajnih zaključaka:

- **Učinkovitost:** ACO se pokazao kao najefikasnija metoda, sa stopom uspešnosti od 100%, što ukazuje na njegovu sposobnost da pronalazi rešenja u svim testiranim slučajevima. S druge strane, BRUT metoda je imala najduže vreme izvršavanja, što je ukazalo na njenu nepraktičnost za veće grafove.
- **Brzina:** GA i VNS su pokazali slične performanse u pogledu vremena izvršavanja, ali VNS je imao nešto bolju stopu uspešnosti. Ovo sugerise da VNS može biti bolji izbor u situacijama gde je brzina važnija od savršenstva rešenja.
- **Fleksibilnost:** Heurističke metode, kao što su GA i ACO, pružaju veću fleksibilnost u primeni na različite tipove grafova i mogu se lako prilagoditi specifičnim zahtevima problema.
- **Praktična primena:** Naši rezultati sugerisu da se heuristički pristupi mogu uspešno koristiti u praksi za rešavanje problema minimalnog nezavisnog dominantnog skupa čvorova, posebno u situacijama gde su klasične metode previše zahtevne ili neefikasne.
- **Preporuke za budući rad:** Istraživanje može biti prošireno uključivanjem drugih heurističkih metoda i optimizacija, kao i ispitivanjem njihovih kombinacija za još bolje rezultate.

Zaključno, ovaj rad ukazuje na značaj heurističkih pristupa u rešavanju složenih problema u teoriji grafova, kao i na potrebu za daljim istraživanjem i razvojem ovih metoda.