Optimizacija – Minimum Number Of Satisfiable Formulas

Nikola Belaković

Sadržaj

1. Uvod

U ovom radu rešavaćemo problem Minimum Number of Satisfiable Formulas koristeći nekoliko različitih tehnika.

Problem spada u NP-teške probleme i izveden je iz problema SAT (problem zadovoljivosti neke formule), koji je jedan od najpoznatijih i najčešće rešavanih NP-teških problema. SAT je i prvi problem za koji je dokazano da je NP-kompletan, pomoću Kukove teoreme.

Svaka od tehnika koje su korišćene u ovom radu (VNS, genetski algoritam, algoritam grube sile i pohlepna pretraga) biće opisana u nastavku.

2. Opis problema

Zadatak problema Minimum Number of Satisfiable Formulas je pronalazak valuacije zadatog skupa promenljivih za koji je zadovoljen najmanji broj formula. Kao ulaz u problem je dat skup promenljivih koji se koriste u formulama i skup formula, a cilj nam je pronaći valuaciju (funkciju koja pridružuje istinitosnu vrednost promenljivim) za koju važi da zadovoljava najmanji broj formula iz datog skupa.

U ovom slučaju svaka od formula će biti u 3CNF obliku, u konjuktivnoj normalnoj formi i sa tačno 3 literala u svakoj klauzi. Konjuktivna normalna forma neke iskazne formule je oblika $A_1 /\backslash A_2 /\backslash \ldots /\backslash A_n$ pri čemu je svaka od formula A_i klauza (disjunkcija literala).

3. Opis algoritama

Za svaki algoritam u ovom radu su korišćene dve pomoćne funkcije. Jedna od njih broji koliko formula u kolekciji je zadovoljeno za određenu valuaciju promenljivih. Ona prolazi kroz sve formule u kolekciji i proverava da li je formula zadovoljena tako što poziva drugu pomoćnu funkciju koja prolazi kroz klauze date formule i proverava da li su sve klauze zadovoljene. To radi tako što iterira kroz sve literale u klauzi, određuje vrednost svake promenljive prema datoj valuaciji i proverava da li je barem jedan literal u klauzi tačan (True). Ako jeste, klauza je zadovoljena.

3.1. Brute-force algoritam

U algoritmu grube sile se koristi metoda *product* iz *itertools* biblioteke, koja će generisati sve moguće valuacije promenljivih, gde za svaku promenljivu može biti postavljena vrednost True ili False. Nakon toga se prolazi kroz sve moguće valuacije i korišćenjem pomoćne funkcije računa broj zadovoljenih formula i poredi sa minimalnim brojem zadovoljenih koji se na početku inicijalizuje na veličinu kolekcije formula. Algoritam će vratiti najmanji broj zadovoljenih formula i valuaciju za koju se taj broj postiže.

Ovaj pristup može biti efikasan samo za male instance, ali će biti sve manje efikasan sa povećanjem broja promenljivih i formula, jer će broj valuacija eksponencijalno rasti. Algoritam garantuje pronalazak najboljeg rešenja jer ispituje sva moguća rešenja.

3.2. Pohlepni algoritam

Pohlepni algoritam je najbrži od svih korišćenih algoritama, zbog svoje pohlepnosti postoje slučajevi kada se on zaglavi u nekom lokalnom optimumu i ne uspe da pronađe globalno najbolje rešenje.

U ovom algoritmu se prolazi redom kroz sve promenljive iz datog skupa, za svaku od njih se u tekućoj valuaciji postavlja vrednost True i proverava koliko je formula zadovoljeno ovom valuacijom. Ako je broj zadovoljenih formula manji od trenutnog minimuma, ažurira se vrednost minimuma i pamti se trenutna valuacija. Zatim se ista stvar radi samo se vrednost promenljive postavlja na False. Na kraju algoritam vraća najmanji broj zadovoljenih formula i valuaciju za koju se taj broj dostiže.

Tačnost rešenja ovog algoritma zavisi od početne vrednosti valaucije i redosleda promenljivih koje se razmatraju. Jedan od načina za povećanje verovatnoće pronalaska boljeg rešenja je veći broj primene algoritma sa različitim početnim vrednostima valuacije i različitim redosledom prolaska kroz promenljive.

3.3. VNS (Variable Neighborhood Search)

Metoda promenljivih susedstva, koju su predložili Hansen i Mladenović 1997. godine, je metaheuristički metod za rešavanje skupa kombinatornih optimizacionih problema. Istražuje promenljive susedstva trenutnog rešenja i prelazi na novo rešenje ako je ono bolje od trenutnog. Koristi metodu lokalne pretrage da bi od rešenja u okruženju došao do lokalnog optimuma.

Jedno od bitnih delova ovog algoritma je metoda shaking (razmrdavanje) koja omogućava da se istraže okoline koje su udaljene od trenutne oblasti koju istražujemo i tako se smanjuje verovatnoća da dobijemo lokalni optimum umesto globalnog.

```
Input: a set of neighbourhood structures N_l, l = 1,2, ..., l_max
S = Initial solution ();
Repeat
        l = 1;
        While (l ≤ l_max)
        {
            S' = Shaking (S, N_l)
            S'* = local search (S')
            if f(S'*) < f(S)
                S ← S'*
                l = 1 ;
            else
                l = l + 1;
        }
Until Stopping criterion is met;

Report: The obtained solution with the lowest f(S)
```

Ova slika pseudokoda nam pokazuje da kvalitet rešenja VNS algoritma zavisi od inicijalizacije početnog rešenja, lokalne pretrage, shakinga.

Inicijalizacija

Za početnu valuaciju je uzeta nasumično generisana valuacija koja sa podjednakom verovatnoćom svakoj promenljivoj dodeljuje jednu od vrednosti True ili False.

Shaking

Funkcija shaking prima kao parametre trenutnu valuaciju i broj k koji predstavlja broj promenljivih iz valuacije za koje će se invertovati vrednost (True prelazi u False, i obrnuto). Ovom metodom generišemo novu, susednu valuaciju.

Lokalna pretraga

U funkciji lokalne pretrage se polazi od trenutne valuacije i zatim se menja vrednost svake promenljive i proverava koliko formula je zadovoljeno korišćenjem pomoćnih funkcija. Ako novi sused ima manji broj zadovoljenih formula od trenutne valuacije, vrednost se ažurira. Proces se ponavlja dok god možemo pronaći bolje susedno rešenje.

Metoda *vns* koja implementira VNS algoritam na početku vrši inicijalizaciju početne valuacije i broji koliko je formula zadovoljeno. Zatim, u glavnoj petlji, algoritam primenjuje strategiju susedstva sa različitim vrednostima parametra k. Takođe, postoji i verovatnoća za prihvatanje lošijeg rešenja, što služi za istraživanje drugih regiona pretrage.

3.4. Genetski algoritam

Genetski algoritam (Genetic algorithm-GA) je jedan od evolutivnih algoritama inspirisanih Darvinovom teorijom evolucije, koja govori o tome da unutar jedne populacije najčešće opstaju najbolje prilagođene jedinke (gde se prilagođenost određuje pomoću fitness funkcije). Dobar je za rešavanje kombinatornih problema. Razvijen je 70-ih godina prošlog veka u Americi od strane J. Hollanda, K. DeJonga i D. Goldberga.

```
Иницијализуј популацију;
Евалуирај популацију; //израчунавање фитнеса хромозома
while Није задовољен услов за завршетак
{
    Одабери родитеље за укрштање;
    Изврши укрштање и мутацију;
    Евалуирај популацију;
}
```

Inicijalizacija

Za početnu valuaciju je uzeta nasumično generisana valuaciju koja sa podjednakom verovatnoćom svakoj promenljivoj dodeljuje jednu od vrednosti True ili False.

Selekcija

Selekcija predstavlja izbor jedinki koje će učestvovati u ukrštanju i ostaviti svoje potomstvo. U selekciji najčešće učestvuju najkvalitetnije jedinke. Postoji nekoliko vrsta selekcija među kojima su ruletska i turnirska. U ovom radu je korišćena turnirska selekcija.

Funkciji za selekciju prosleđujemo populaciju jedinki (svaka jedinka ima svoju fitness vrednost koja predstavlja broj zadovoljenih formula i koja se računa

koristeći pomoćnu funkciju), broj jedinki koji će učestvovati u turniru i indeks jedinke koja je zabranjena da bude odabrana tokom selekcije. Nasumično se bira skup jedinki i najbolja od njih se bira za selekciju.

Mutacija

Mutacija predstavlja promenu nekih gena određene jedinke. Jedan je od osnovnih genetskih operatora koji služi za uvođenje raznolikosti u populaciji, čime se omogućava istraživanje novih delova prostora pretrage i izbegavanje zaglavljivanja u lokalnom optimumu. Za svaku jedinku se primenjuje sa nekom malom verovatnoćom. U ovom radu je korišćena mutacija zasnovana na izvrtanju bita sa fiksnom verovatnoćom.

Funkciji za mutaciju se prosleđuje jedinka iz populacije i verovatnoća da će doći do mutacije. Funkcija prolazi kroz sve promenljive u jedinki, koja predstavlja valuaciju, i invertuje joj vrednost ako je nasumično generisan broj od 0 do 1 manji od parametra verovatnoće koji smo prosledili.

Ukrštanje

Ukrštanje predstavlja ključni operator koji se koristi za kombinovanje genetskog materijala dve jedinke kako bi se stvorila nova potomstva.

U korišćenom algoritmu ukrštanja se prvo bira pozicija na kojoj se vrši ukrštanje. Zatim se u prvo dete kopira genetski kod prvog roditelja do te pozicije i ista stvar za drugo dete. Nakon toga se od izabrane pozicije u prvo dete kopira genetski kod drugog roditelja i analogno za drugo dete.

U implementiranom genetskom algoritmu je korišćena tehnika nazvana elitizam koja se zasniva na ideji da se najbolja ili više najboljih jedinki direktno prenese u narednu generaciju.

4. Rezultati

Algoritmi su testirani na random generisanim podacima.

Prvi skup podataka: *random_3cnf_small_test_1.txt* ima 8 promenljivih i 27 formula sa različitim brojem klauza (od 1 do 3). Nad ovim test skupom je pokrenut i brute force algoritam jer je skup podataka malih dimenzija pa je vreme izvršavanja prihvatljivo i za ovaj algoritam. Svi korišćeni algoritmi su u ovom slučaju pronašli optimalno rešenje, koje je isto kao i rešenje dobijeno brute force algoritmom što pokazuje da metode rade ispravno.

Dobijeni rezultati:

1. Brute force: 19
2. Greedy search: 19
3. VNS: 19
4. Genetski algoritam: 19

Pored rešenja (broja minimalno zadovoljenih formula), na slici 1 je prikazana i valuacija promenljivih za koju se dobija dato rešenje.

```
Brute force results:
Minimum number of satisfiable formulas:  19
Valuation for minimum number of satisfiable formulas:  {1: True, 2: Tr
ue, 3: True, 4: True, 5: False, 6: False, 7: False, 8: False}

Brute force results:
Minimum number of satisfiable formulas:  19
Valuation for minimum number of satisfiable formulas:  {1: True, 2: Fa
lse, 3: True, 4: True, 5: False, 6: False, 7: False, 8: False}

Variable Neighborhood Search results:
Minimum number of satisfiable formulas:  19
Valuation for minimum number of satisfiable formulas:  {1: True, 2: Fa
lse, 3: False, 4: False, 5: False, 6: False, 7: False, 8: False}

Genetic algorithm results:
Minimum number of satisfiable formulas:  19
Valuation for minimum number of satisfiable formulas:  {1: False, 2: F
alse, 3: True, 4: False, 5: False, 6: False, 7: True, 8: False}
```

Slika 1

Drugi skup podataka: *random_3cnf_small_test_1.txt* ima 8 promenljivih i 22 formule sa različitim brojem klauza (od 1 do 3). Nad ovim test skupom je pokrenut i brute force algoritam jer je skup podataka malih dimenzija pa je vreme izvršavanja prihvatljivo i za ovaj algoritam. Svi korišćeni algoritmi su u ovom slučaju pronašli optimalno rešenje, koje je isto kao i rešenje dobijeno brute force algoritmom što pokazuje da metode rade ispravno.

Dobijeni rezultati:

5. Brute force: 12
6. Greedy search: 12
7. VNS: 12
8. Genetski algoritam: 12

Pored rešenja (broja minimalno zadovoljenih formula), na slici 2 je prikazana i valuacija promenljivih za koju se dobija dato rešenje.

```
Brute force results:
Minimum number of satisfiable formulas:  12
Valuation for minimum number of satisfiable formulas:  {1: False, 2: T
rue, 3: False, 4: False, 5: False, 6: True, 7: False, 8: True}

Brute force results:
Minimum number of satisfiable formulas:  12
Valuation for minimum number of satisfiable formulas:  {1: False, 2: T
rue, 3: False, 4: False, 5: False, 6: True, 7: False, 8: True}

Variable Neighborhood Search results:
Minimum number of satisfiable formulas:  12
Valuation for minimum number of satisfiable formulas:  {1: False, 2: T
rue, 3: False, 4: False, 5: False, 6: False, 7: False, 8: True}

Genetic algorithm results:
Minimum number of satisfiable formulas:  12
Valuation for minimum number of satisfiable formulas:  {1: False, 2: T
rue, 3: False, 4: False, 5: False, 6: False, 7: False, 8: True}
```

Slika 2

Treći skup podataka: *random_3cnf_large_test_3.txt* ima 18 promenljivih i 121 formulu sa različitim brojem klauza (od 1 do 10). Nad ovim test skupom nije pozvana metoda brute force jer je skup preveliki i metoda ne završava svoj rad u razumnom vremenu. Rezultati prikazani u tabeli 1.

Svaka metoda je pozvana 5 puta i zabeleženo je ukupno vreme izvršavanja tih 5 iteracija (**total time**), prosečno vreme izvršavanja jedne iteracije (**avg time**), prosečna rešenja dobijena kao rezultat izvršavanja metode (**avg value**) i najbolje rešenje dobijeno tokom tih 5 iteracija (**best value**).

Method name	Total time(sec)	Average time(sec)	Average value	Best value
Greedy search	0.024	0.005	53.0	**47**
VNS	2.794	0.559	45.0	**45**
Genetic algorithm	0.720	0.144	46.4	**45**

Tabela 1

Četvrti skup podataka: *random_3cnf_large_test_4.txt* ima 33 promenljivih i 206 formula sa različitim brojem klauza (od 1 do 10). Nad ovim test skupom nije pozvana metoda brute force jer je skup preveliki i metoda ne završava svoj rad u razumnom vremenu. Rezultati prikazani u tabeli 2.

Method name	Total time(sec)	Average time(sec)	Average value	Best value
Greedy search	0.083	0.017	78.8	**74**
VNS	9.387	1.877	66.0	**65**
Genetic algorithm	1.350	0.270	68.4	**65**

Tabela 2

Peti skup podataka: *random_3cnf_large_test_5.txt* ima 49 promenljivih i 352 formule sa različitim brojem klauza (od 1 do 10). Nad ovim test skupom nije pozvana metoda brute force jer je skup preveliki i metoda ne završava svoj rad u razumnom vremenu. Rezultati prikazani u tabeli 3.

Method name	Total time(sec)	Average time(sec)	Average value	Best value
Greedy search	0.194	0.039	146.6	**139**
VNS	22.069	4.414	126.8	**125**
Genetic algorithm	2.426	0.485	142.4	**138**

Tabela 3

Šesti skup podataka: *random_3cnf_large_test_6.txt* ima 40 promenljivih i 211 formula sa različitim brojem klauza (od 1 do 10). Nad ovim test skupom nije pozvana metoda brute force jer je skup preveliki i metoda ne završava svoj rad u razumnom vremenu. Rezultati prikazani u tabeli 3.

Method name	Total time(sec)	Average time(sec)	Average value	Best value
Greedy search	0.118	0.024	87.2	**82**
VNS	10.395	2.079	73.4	**72**
Genetic algorithm	1.386	0.277	82.0	**77**

Tabela 4

5. Zaključak

U ovom radu rešavan je problem Minimum Number of Satisfiable Formulas koristeći nekoliko različitih tehnika. Cilj problema je pronalazak valuacije zadatog skupa promenljivih za koji je zadovoljen najmanji broj formula. Problem spada u NP-teške probleme i izveden je iz problema SAT.

Problem je rešavan koristeći nekoliko metoda: Metoda grube sile, Genetski algoritam, Metoda promenljivih okolina (VNS metoda) i Metoda pohlepne pretrage.

Iz dobijenih rezultata mogu se izvesti neki zaključci. Zbog eksponencijalne složenosti metode grube sile, vidimo da ne uspeva da završi rad u razumnom vremenu za probleme sa više od 15 promenljivih. Zbog toga je gruba sila poslužila kao merilo kvaliteta ostalih metoda na jako malim skupovima podataka, na kojima smo videli da su se svi algoritmi dobro pokazali i pronašli optimalno rešenje.

Sa povećanjem broja promenljivih i broja formula problem se komplikuje i postaje sve teži za brzo i optimalno rešavanje. Pohlepni algoritam se u problemima velikih dimenzija pokazuje kao najbrži, jer je najmanje složenosti od svih korišćenih metoda. Njegov problem je što tačnost rešenja zavisi od početne vrednosti valuacije i redosleda promenljivih koje se razmatraju, što dovodi često do rešenja koje nije dovoljno dobro.

VNS algoritam vraća najbolja rešenja za velike skupove podataka, ali je njegovo vreme izvršavanja dosta veće nego kod pohlepnog i genetskog algoritma.

Genetski algoritam vraća malo lošija rešenja od VNS, ali se izvršava dosta brže i zato u nekim situacijama može biti bolji izbor.

6. Literatura

1) https://poincare.matf.bg.ac.rs/~denis.alicic/ri/6.html
2) http://poincare.matf.bg.ac.rs/~aleksandar.kartelj/nastava/RI2022/06.Genetski.algoritmi.pdf
3) https://en.wikipedia.org/wiki/Variable_neighborhood_search
4) https://www.researchgate.net/figure/The-pseudo-code-of-the-proposed-VNS_fig7_325559857
5) https://github.com/MATF-RI/Materijali-sa-vezbi