Prirodno-matematički fakultet
Odsjek za matematiku
Teorijska kompjuterska nauka
Napredni algoritmi i strukture podataka

Dokumentacija za projekat 2

Implementacija algoritama Simultano kaljenje i Tabu Pretraga za SAT problem

KASUMOVIĆ MUAZ

Sarajevo, juni 2019.

Sadržaj

Uvod

U ovom radu će biti opisani detalji implementacije metaheuristika "Tabu-Search" i Simultano kaljenje za rješavanje SAT problema (engl. boolean satisfability problem). Algoritmi su implementirani u programskom jeziku C++. Također, u okviru projekta implementirana je i greedy heuristika za rješavanje SAT problema i dato poređenje sa navedenim metaheuristikama. Algoritmi su testirani na instancama problema različitih veličina te su rezultati prikazani na kraju rada.

SAT problem

Problem zadovoljenja logičke formule (engl. boolean satisfability problem or propositional satisfability problem), ili skraćeno SAT, je problem određivanja postojanja dodjele logičkim varijablama tako da je logička funkcija zadovoljena. Koristeći Bool-ovu algebru svaka logička funkcija se može svesti na konjuktivno normalnu formu, ili skraćeno CNF. CNF se sastoji od konjunkcije jedne ili više klauzula, a klauzula je disjunkcija literala pri čemu je literal varijabla ili negacija varijable.
Provjera zadovoljenja logičke formule date u CNF-u je problem od temeljne važnosti u teoriji kao i u praksi u kompjuterskoj nauci. SAT je bio prvi problem za koji se dokazalo da je NP-kompletan, što znači da nema poznat efikasan algoritam za njegovo rješavanje. Svi poznati algoritmi za NP-kompletne probleme imaju eksponencijalnu vremensku složenost u najgorem slučaju. Doslovno hiljade problema, od kojih su mnogi problemi od velike praktične važnosti, su NP-kompletni. Klasa NP-kompletnih problema ima neobičnu karakteristiku da ako se pronađe efikasan algoritam za jedan problem (algoritam čije vrijeme izvršavanja je polinomijalno) tada imamo i efikasne algoritme za ostale probleme iz klase. To znači da reprezentacija i algoritam jednog problema (npr. SAT) se mogu iskoristiti za ostale probleme iz klase NP-kompletnih problema.
Trenutno nije poznat algoritam koji učinkovito rješava svaki SAT problem, a općenito se smatra da takav algoritam ne postoji; ipak, ovo uvjerenje nije dokazano matematički, a rješavanje pitanja ima li SAT polinomijalni algoritam ekvivalentno je problemu P vs NP, što je poznati otvoreni problem u teorijskoj kompjuterskoj nauci.
Prostor rješenja ima 2^n mogućih rješenja što već za male instance problema onemogućava pretragu cijelog prostora. Pošto već za male vrijednosti n pretrage cijelog prostora nije moguća razvijeni su brojni nekompletni algoritmi koji se zasnivaju na različitim heurisitikama. U narednim dijelovima dokumentacije će biti predstavljeni greedy algoritam, simultano kaljenje i tabu pretraga.

Reprezentacija SAT problema

U ovom dijelu će biti opisana reprezentacija SAT problema koja je korištena u okviru projekta. Razvijena je klasa Formula koja sadrži tri atributa: broj varijabli, broja klauzula i klauzule. Logička formula se učitava iz .cnf fajla u kojem se nalazi formula zapisana u CNF formi. Fajl je organizovan na način da se u prvom redu nalazi broj varijabli i broj klauzula, a u svakom sljedećem redu se nalazi klauzula. Primjer klauzule je: 10 11 -15 što znači da ovaj klauzula ima sljedeći oblik $x_{10} \vee x_{11} \vee \neg x_{15}$. Klauzule su organizovane kao vektor vektora cijelih brojeva. Klasa Formula ima dvije metode članice:

- *int prebrojZadovoljeneKlauzule (vector<bool> dodjela)* i

- *void ispisiFormulu()*.

Prva funkcija za proslijeđenu dodjelu vraća broj klauzula koje su zadovoljene. Druga funkcija ispisuje formulu tako da u svaki red ispisuje novu klauzulu.

Greedy SAT algoritam

Kao što je već kazano da za veće instance problema nije moguće koristiti potpunu pretragu prostora rješenja (tzv. kompletne algoritme) pribjegava se korištenju nekompletnih algoritama. Greedy SAT algoritam (GSAT) je nekompletan algoritam, što znači da ovaj algoritam ne garantuje da će moći dati tačan odgovor da li postoji dodjela koja zadovoljava logičku funkciju, ali vrijeme izvršavanja ovog algoritma se može kontrolisati. Nekompletni algoritmi su tipično stohastičke tehnike (tj. bazirane na randomizaciji) lokalne pretrage. Najistaknutiji među ovim algoritmima su greedy algoritmi.
U osnovi, GSAT algoritam počinje s slučajnom dodjelom logičkih konstanti (true i false) T za datu formula S. Ako ova dodjela zadovaljava S, vraća se T. U suprotnom, GSAT odabire sljedeću dodjelu T. T zadovoljava samo određeni podskup klauzula u S. Nova dodjela T je izabrana tako da zadovaljava najviše klauzula od svih susjeda početne dodjele. Odabiranje sljedeće dodjele vrši se MAX_FLIPS puta. Parametar MAX_TRIES definiše koliko će se puta generisati slučajna dodjela i izvršiti pretraživanje njene okoline. GSAT će pokušati naći dodjelu koja zadovaljava logičku formulu u $MAX\_TRIES * MAX\_FLIPS$ iteracija. Na ovaj način je definisano vrijeme izvršavanja algoritma.

Algorithm 1: GSAT algoritam

Input : Formula u CNF - S, MAX-TRIES i MAX-FLIPS
Output: Pronađena dodjelan, N- najbolja pronađena dodjela

```
for i ← 1 to MAX_TRIES do
    T := random dodjela varijablama
    N := T
    for j ← 1 to MAX_FLIPS do
        if T zadovoljava S then
            return true
        end
        if T bolje od N then
            N := T
        end
        T := generiši sljedeću najbolju dodjelu iz okoline
    end
end
return false
```

Prototip funkcije koja implementira ovaj algoritam je
bool GSAT(const Formula& f, int maxTries, int maxFLips, vector<bool> & dodjela, int& broj_tacnih).

Funkcija prima po referenci vektor dodjele logičkih vrijednosti varijablama i broj tačnih klauzula u formuli za datu dodjelu. Zbog toga što se ovi parametri šalju po referenci funkciji oni služe kao povratne vrijednosti. Dakle nakon izvršavanja funkcije vratit ćemo tri vrijednosti: da li je pronađena dodjela koja zadovoljava logičku formulu, sama dodjela i broj tačnih klauzula u formuli.

Tabu Search

Tabu search je metaheuristička metoda pretrage koja je bazirana na metodi lokalne pretrage. Lokalna pretraga uzima potencijalno rješenje problema i provjerava njegove neposredne susjede (to jest, rješenja koja se nalaze u okolini datog rješenja) u nadi da će pronaći bolje rješenje. Poznato je da metode lokalne pretrage imaju tendenciju da se zaglavljuju u lokalnim ekstremima.
Tabu search poboljšava izvedbu lokalne pretrage izmjenom osnovnog pravila lokalne pretrage na način da se u svakom koraku može prihvatiti i lošije rješenje ukoliko nije dostupno bolje rješenje. Osim toga, zabrane (tj. tabu) su uvedene kako bi 'obeshrabrile' pretragu da se vrati na prethodno posjećena rješenja. Ovakvim pristupom se omogućava bijeg iz lokalnog optimuma.

Za SAT problem definisanje zabranjenih rješenja se može postići na način da se varijabli čija je logička vrijednost promijenjena u k-toj iteraciji 'zabrani' da bude mijenjana u narednih TABU iteracija. Da bi se uspjelo voditi računa koja rješenja su zabranjena (tabu) potrebno je imati memorijsku strukturu u kojoj će se voditi evidencija o zabranjenim rješenjima. Za ovaj problem memorijska struktura je niz čija je veličina jednaka broju varijabli. U početku se elementi niza inicijaliziraju nulama. U k-toj iteraciji iz trenutnog rješenja samo se može otići u susjedna rješenja koja su dostupna (odnosno samo se mogu 'flipovat' varijable čija tabu vrijednost je nula). Nakon k-te iteracije svim varijablama se smanji vrijednost u tabu nizu, tj. smanji se broj narednih iteracija u kojima varijabla neće biti dostupna za 'flipovanje', dok se varijabli koja je promijenila logičku vrijednost postavi vrijednost u tabu nizu na TABU (koliko narednih iteracija je nedostupna za izmjenu). Pseudokod je dat listingom ispod.

Algorithm 2: Tabu Search algoritam

Input : Formula u CNF - S, TABU, broj-iteracija
Output: Pronađena dodjela, N - najbolje pronađeno rješenje

```
inicijaliziraj tabu niz
T := random dodjela varijablama
N := T
for i ← 1 to broj-iteracija do
    if T zadovoljava S then
        return true
    end
    if T bolje od N then
        N := T
    end
    T := najbolje rješenje u okolini trenutnog koje se dobija
    'flipovanjem' jedne varijable koja nije zabranjena
    ažuriraj tabu niz
end
return false
```

Prototip funkcije koja implementira ovaj algoritam je
bool TabuSearch (const Formula& f, int broj_iteracija,int TABU,vector<bool> & dodjela, int&

broj_tacnih_klauzula).

Funkcija prima po referenci vektor dodjele logičkih vrijednosti varijablama i broj tačnih klauzula u formuli za datu dodjelu. Zbog toga što se ovi parametri šalju po referenci funkciji oni služe kao povratne vrijednosti. Dakle nakon izvršavanja funkcije vratit ćemo tri vrijednosti: da li je pronađena dodjela koja zadovoljava logičku formulu, sama dodjela i broj tačnih klauzula u formuli.

Simultano kaljenje

Simultano kaljenje (engl. simulated annealing) je poznata heuristička tehnika optimizacije koja oponaša prirodni proces sporog hlađenja tekućina što dovodi do čvrstog oblika koji ima najnižu energiju. U biti vjerovatnoća na temperaturi T s kojom će se preći iz stanja S_1, sa energijom E_1, u stanje S_2, sa energijom E_2, data je sa $1/(1 + e^{(E1-E2)/kT})$. Interesantna stvar je da ovakva distribucija dozvoljava da algoritam (koji traži stanje sa najmanjom energijom) pređe i u stanje sa većom energijom (slabije rješenje) tokom pretrage, ali sa malom vjerovatnoćom. Ova vjerovatnoća se smanjuje sa smanjivanjem temperature. Simultano kaljenje je također nekompletan algoritam, kao i GSAT i Tabu Search.

Da bi se primjenio algoritam simultanog kaljenja potrebno je definisati energiju i stanje. Stanje odgovara nizu varijabli kojima je dodjeljena logička vrijednost dok se energija može definisati kao broj zadovoljenih klauzula u formuli. Razlika energija je jednostavno razlika broja zadovoljenih klauzula u dvije dodjele. Boltzmanova konstanta k se uzima da je 1. Pojam temperature često je čisto umjetan(to jest, ne odgovara niti jednom stvarnom aspektu problema) i može se definirati da varirati od zadane maksimalne vrijednosti T_{max} do zadane minimalne temperature T_{min}. Koristi se nekoliko strategija hlađenja koje sistematično spuštaju temperaturu od T_{max} do T_{min}. Najčešća strategija koja se primjenjuje je strategija po kojoj se temperature smanjuje po zakonu geometrijske serije, tj.

$$T_{sljedeca} = \alpha * T_{trenutna} \tag{1}$$

pri čemu je α zadana konstanta između 0 i 1.

Algoritam je implementiran na način da se za unaprijed zadan broj iteracija generiše random dodjela. Dalje, trenutno rješenje se mijenja na sljedeći način: prolazi se kroz sve varijable i negira se logička vrijednost sa vjerovatnoćom $1/(1 + e^{\delta/kT})$, pri čemu je δ razlika tačnih klauza u trenutnoj dodjeli i dodjeli u kojoj je promijenja j-ta varijabla. Ovaj postupak se ponavlja dok se sistem ne ohladi, tj. dok temperatura ne dostigne vrijednost T_{min}. Temperatura se smanjuje po zakonu definisanom relacijom 1.

Jasno je da će izbor parametata podrazumijevati određene kompromise. Za zadani maksimalni broj iteracija, smanjujući T_{min} i / ili povećavajući T_{max} omogućit će više pokušaja za svaku iteraciju što će dovesti do ukupnog povećanja vremena izvršavanja algoritma. Slična situacija se događa i ako smanjimo ili povećamo koeficijent α. Povećanjem temperaturnog raspona (ili smanjenjem koeficijenta α) smanjujemo broj neovisnih pokušaja koji će završit u prihvatljivom vremenu, ali temeljitije istražujemo prostor tokom svakog pokušaja. Situacija je obrnuta ako se temperaturni raspon smanji (ili povećava koeficijent α). Nažalost, uopćeno nije jasno da li je bolje napraviti više neovisnih pokušaja ili temeljitije tražiti tokom svakog pokušaja. Iskustvo igra veliku ulogu u

odabiru ovih parametara. Pseudokod algoritma je dat listingom ispod.

Algorithm 3: Simultano kaljenje

Input : Formula u CNF - S, T_{max}, T_{min}, α,MAX-TRIES
Output: Pronađena dodjela, N - najbolje pronađeno rješenje

```
for i ← 1 to MAX-TRIES do
    R := random dodjela varijablama
    N := R
    while T > T_min do
        if R zadovoljava S then
            return true
        end
        if R bolje od N then
            N := T
        end
        for k ← 1 to broj_varijabli do
            izračunaj δ ako se varijabla v_k negira
            negiraj varijablu v_k sa vjerovatnoćom 1/(1 + e^{δ/kT})
            ažuriraj R ako se desilo negiranje
        end
    end
    T := T_max
end
return false
```

Prototip funkcije koja implementira ovaj algoritam je
bool SimultanoKaljenje (const Formula& f,vector<bool> & dodjela, int& broj_tacnih_klauzula, double Tstart, double Tmin, double alpha, int MAXTRIES).

Rezultati i zaključak

Nakon implementiranja algoritama provedeno je testiranje te njihova usporedba. Algoritmi su testirani na instancama različitih veličina sa različitim parametrima algoritama. Rezultati su prikazani u tabelama 1 i 2. Testiranje je izvršeno na računaru sa karakteristikama:

- Procesor: Intel(R) Core(TM) i5-3320M CPU @2.60Ghz 2.60Ghz,
- RAM: 4.00 GB,
- Operativni sistem: Windows 10.

U tabeli 1 su prikazani rezultati izvršavanja algoritma simultanog kaljenja. Može se zaključiti da je ovaj algoritam dao malo bolje rezultate na pojedinim instancama u odnosu na Tabu Search. U odnosu na GSAT simultano kaljenje je dalo bolje rezultate, što se posebno primjećuje na većim instancama.

Iz priloženih rezultata u tabeli 2 se može zaključiti da na testiranim instancama greedy algoritam nije dao nijednom bolji rezultat od Tabu Search-a. Na instancama manje veličine GSAT i Tabu Search daju iste rezultate što je očekivano jer prostor pretrage nije ogroman pa GSAT istraži veliki dio prostora. Povećanjem vrijednosti TABU (koliko narednih iteracija je zabranjeno negirati varijablu) dobija se veći broj zadovoljenih klauzula. Važno je napomenuti da je broj iteracija za GSAT algoritam znatno veći od broja iteracija Tabu Search-a.

Može se zaključiti da je za dobre rezultate potrebno pronaći dobre parametre koji se proslijeđuju algoritmi. Određivanje ovih parametara nije jednostavno i potrebno je dosta iskustva u tom procesu.

Broj varijabli	**Broj klauzula**	Tstart =10 Tmin=0.1 alpha=0.01 MAXTRIES=5	Tstart =100 Tmin=0.1 alpha=0.05 MAXTRIES=10	Tstart =3 Tmin=0.1 alpha=0.01 MAXTRIES=20
100	160	159	159	159
50	80	79	79	79
42	133	132	132	132
100	2808	2799	2802	2804
250	1065	1062	1058	1063
225	960	958	956	957
200	860	858	857	859
100	430	427	428	428
179	2928	2916	2917	2919

Tablica 1: Rezultati testiranja simuliranog kaljenja

Broj varijabli	**Broj klauzula**	**GSAT** MAX-TRIES=5 MAX-FLIPS=1000	**Tabu Search** TABU = 10 BR-ITER=1000	**GSAT** MAX-TRIES=10 MAX-FLIPS=500	**Tabu Search** TABU = 15 BR-ITER = 1000	**GSAT** MAX-TRIES=1 MAX-FLIPS=1000	**Tabu Search** TABU = 17 BR-ITER = 1000
100	160	157	159	158	159	155	159
50	80	78	78	78	78	78	79
42	133	132	132	132	132	132	132
100	2808	2799	2802	2805	2806	2797	2804
250	1065	1040	1043	1045	1060	1037	1054
225	960	940	946	941	950	935	950
200	860	841	852	847	857	834	856
100	430	422	426	422	426	421	428
179	2928	2908	2915	2908	2918	2904	2920

Tablica 2: Rezultati testiranja GSAT i Tabu Search algoritama