Univerzitet u Sarajevu
Prirodno-matematički fakultet
Odsjek za matematiku

Napredni algoritmi i strukture podataka

Projekat 2

Chahin Malek

Uvod

U ovom radu izvršena je komparacija tradicionalnih algoritama za rješavanje SAT problema i metaheurističkih algoritama *simulirano kaljenje* i *tabu search*. Pored toga, data je implementacija spremanja logičkih formula u konjuktivnoj normalnoj formuli u memoriji računara. Implementirani su algoritmi koji rješavaju problem MAX-SAT koji predstavlja generalizaciju SAT problema i čijim bi rješavanjem bili u stanju riješiti i SAT problem jednostavno radeći provjeru *MAX-SAT(F) = CLAUSE(F)*.

Arhitektura projekta

Projekat se sastoji od klase **CNF** koja predstavlja način pohrane formule u konjuktivnoj normalnoj formi u memoriji računara i pomoćnih ili testnih fajlova koji su autoru služili u svrhu testiranja spomenutih alogritama. Pored ovih fajlova, u direktoriju *data* se nalaze sve instance na kojima su pokrenuti algoritmi, odnosno, instance koje su služile za komparaciju algoritama. Sve instance dostupne su online na sljedećoj stranici. Ukoliko je potrebno ubaciti nove instance u postojeći projekat, tada ih je potrebno spremiti u formatu u kojem su date ostale instance u direktoriju *data*.

Spremanje logičkih formula

Sve logičke formule, u konjuktivnoj normalnoj formi, moguće je spremiti u memoriju računara koristeći arhitekturu datu u nastavku.

1. Svaka varijabla predstavlja pokazivač na jednu od dvije alocirane memorijske adrese *TRUE* i *FALSE*
2. Za predstavljanje jedne logičke formule potrebno je čuvati niz (ili vektor) varijabli
3. Svaka klauza se sastoji od proizvoljnog broja varijabli (>0) i njihovog znaka (negacija)
4. Za predstavljanje jedne logičke formule potrebno je čuvati niz (ili vektor) klauza

Stavka 1., u upravo navedenoj listi, je dizajnirana na način da korisnik može alocirati bilo kakvu vrijednost u pozadini a da je algoritam smatra kao **true** ili **false** vrijednost. Pošto svi algoritmi, koji su predstavljeni u ovom radu, rade isključivo na dodjeljivanju varijablama druge vrijednosti i provjere broja zadovoljenih klauza, to po mišljenju autora, ovakva arhitektura predstavlja idealan pristup jer je samo potrebno preusmjeriti pokazivač, naveden u stavci 1., na memorijsku adresu koja predstavlja suprotnu logičku vrijednost.

Predstavljanje rješenja

Rješenje SAT problema predstavlja bilo koji izbor vrijednosti u nizu koji čuva vrijednosti varijabli. Susjedstvo proizvoljnog rješenja biramo na način da uzmemo sva rješenja koja se razlikuju na samo jednom mjestu (u samo jednoj vrijednosti varijabli) od trenutnog rješenja.

Tradicionalni algoritmi

U ovom dijelu rada je dat opis dva tradicionalna *local search* algoritma koji su greedy prirode i koriste dvije različite heuristike kojima se vode u potrazi za boljim rješenjem.

Lokalna pretraga

Ovaj algoritam, kako mu i ime kaže, posmatra susjedstvo trenutnog rješenja i prihvata rješenje ukoliko ono zadovoljava veći broj klauza. Pseudokod algoritma dat je u nastavku.

```
do
    For each neighbor solution N do
        if N better than current solution then
            keep N
        else
            drop N
while improvements are made
```

Hill climbing

Ideja ovog algoritma jeste, kao i u prethodnom, obići susjedstvo datog rješenja imajući u vidu frekvenciju varijable u klauzama date formule. Ideja je da klauze koje se pojavljuju u većem broju klauza probamo prve zadovoljiti. Ideja algoritma je greedy bazirana i postoje kontra primjeri u kojima ovakav primjer naravno ne daje dobre rezultate. Eksperimentalno je pokazano da je bolje prvo zadovoljiti klauze varijablama koje su manje učestale nego, kako je navedeno, da se to vrši koristeći varijable koje su učestalije. Iz ovog razloga je implementacija ovog algoritma takva da je korisniku ostavljen izbor kriterijske funkcije. U rezultatima su predstavljena rješenja koristeći oba pristupa. Pseudokod ovog algoritma dat je u nastavku.

```
do
    v <- variable that has best frequency with value b
    create solution N by assigning value b to variable v

    if N better than current solution then
        Keep N
    else
        revert move
while improvements are made
```

Ostale ideje

Ideje vrijedne spominjanja prilikom rješavanja SAT problema su slične gore navedenim idejama s razlikom da dopunjuju probleme na koji nailaze ti algoritmi. Jedna od ideja jeste da od svih susjednih rješenja, pored uzimanja u obzir frekvenciju rješenja, posmatramo i dužinu klauze. Intuicija iza ove ideje jeste da su *nezgodnije* za zadovoljiti klauze koje imaju manji broj varijabli u sebi. Pored toga, moguće je zabriniti poteze koji će neku od klauza koja je prethodno bila zadovoljena učiniti nezadovoljenom.

Metaheuristički algoritmi

U ovom dijelu rada date su ideje i pseudokodovi za rješavanje SAT problema koristeći dva algoritma metaheurističke prirode: *tabu search* i *simulirano kaljenje*. Oba algoritma imaju za ideju izbjegavanje lokalnih optimuma nađenih iz inicijalnog rješenja. Moguće je ovim algoritmima kao početno rješenje proslijediti rješenje dobijeno koristeći ideje navedene u prethodnoj oblasti.

Simulirano kaljenje

Stohastički algoritam koji predstavlja balans između *random walk* i *hill climbing* algoritama. Osnovna ideja koja služi ovom algoritmu kada je u pitanju bježanje iz lokalnog optimuma jeste prihvatiti *dovoljno dobro* rješenje s određenom vjerovatnoćom. Ideja je i dalje obići susjedna rješenje, s početka vodeći se idejom *random walk* algoritama a pretragu progresivno usmjeravati ka boljem rješenju kroz iteracije algoritma. Susjedstvo rješenja je isto kao i u prethodno predstavljenim algoritmima međutim zbog ideje na kojoj je zasnovan ovaj algoritam, moguće je obići veći dio prostora pretrage nego što je to moguće koristeći *eksploataciju* rješenja kao što to rade algoritmi navedeni u prethodnom poglavlju. Pseudokod algoritma je dat u nastavku.

```
do
    N <- random neighbor solution
    if N better than current solution then
        keep N
    else
        ΔE = N - currentSolution
        if e^(-ΔE/T) > random() then
           keep N
        else
           drop N
    T <- T * (1 - coolingRate)
while T > 1
```

Prilikom implementacije, korisniku je prepušten izbor hiperparametara koji redom predstavljaju inicijalnu temperaturu kao i stepen hlađenja.

Tabu search

Deterministički algoritam koji, kao i prethodni algoritam, ima za cilj izbjegavanje lokalnih optimuma. Za razliku od prethodnog algoritma, ovaj je algoritam deterministički, dakle za više različitih pokretanja nad istim instancama dobiju se ista rješenja. Ideja ovog algoritma jeste zabriniti potez, odnosno učiniti ga *tabu*, izvjestan broj koraka. Ideja je i dalje pretražiti susjedna rješenja rješenju kojeg trenutno posmatramo i uzeti *najbolje susjedno rješenje* kao sljedeći izbor neovisno od toga da li je ono bolje od trenutnog rješenja ili ne, a zatim zabraniti mijenjanje varijable koja je dovela do novog rješenja izvjestan broj koraka što korisnik treba unaprijed da definiše. Na ovaj način je moguće udaljiti se od rješenja i ne vraćati se u njega a sve vrijeme čuvati najbolje rješenje na koji je algoritam naišao u toku svog rada. Tabu search algoritmi mogu naići na problem radi ovog pristupa, odnosno šta ako se desi slučaj gdje smo se odmakli dovoljno od nekog lokalnog optimuma, mijenjanje varijable *b* je i dalje nedozvoljeno a baš se mijenjanjem te varijable dobije globalni optimum. Ideja u implementaciji jeste dozovoliti *tabu* potez ukoliko su sva susjedna rješenja gora od trenutnog a *tabu* potez dovodi do boljeg rješenja od najboljeg na kojeg ja algoritam naišao dotad. Pseudokod algoritma je dat u nastavku.

```
do
    N <- best amongst other neighbor solutions

    if N worse than current then
       if a tabu move T exists such that T > best then
           keep T
           update restriction for T
           update best
        else
           return best
    else
        make move to N
        restrict move to N

        if N better than best then
            update best
while no better solution found in x steps
```