Stohastički SAT rešavači

Seminarski rad u okviru kursa
Automatsko rezonovanje
Matematički fakultet

Marko Radosavljević, mi241010@alas.matf.bg.ac.rs

Sažetak

Ovaj rad se bavi implementacijom i analizom stohastičkih algoritama za rešavanje problema zadovoljivosti formula u iskaznoj logici, sa posebnim fokusom na GSAT i WalkSAT. Iako ovi algoritmi nude efikasna rešenja za mnoge primene, njihov heuristički karakter znači da ne garantuju uvek pronalaženje rešenja, čak ni za zadovoljavajuće formule. Kroz implementaciju i analizu, rad takođe razmatra mogućnosti za dalja poboljšanja i proširenja ovih tehnika, uključujući optimizaciju kroz kombinaciju metoda i istraživanje novih metaheurističkih pristupa.

1 Uvod

Rešavanje problema zadovoljenja formula u iskaznoj logici (SAT problem) jedan je od centralnih problema u teoriji računarstva. SAT problem je prvi problem za koji je dokazano da je NP-potpun, što znači da pripada klasi problema čija se rešenja mogu proveriti u polinomijalnom vremenu, dok efikasan algoritam za pronalaženje rešenja nije poznat. NP-potpuni problemi su od ključnog značaja u računarstvu jer, ukoliko bi se pronašao algoritam za efikasno rešavanje jednog od njih, svi problemi u klasi NP mogli bi se rešiti efikasno. Zbog ovih osobina, SAT problem ima široku primenu u oblastima kao što su verifikacija softvera i hardvera, planiranje i veštačka inteligencija.

Jedna od ključnih kategorija algoritama za rešavanje SAT problema su stohastički algoritmi, koji koriste metode zasnovane na heuristikama i nasumičnosti kako bi pronašli rešenje. U ovom radu fokus je na dva reprezentativna stohastička pristupa: GSAT (Greedy SAT) i WalkSAT algoritmima. Ovi algoritmi poznati su po svojoj jednostavnosti i efikasnosti u rešavanju velikih formula, posebno u slučajevima kada deterministički algoritmi nisu praktični.

Cilj ovog rada je implementacija ova dva algoritama, uz objašnjenje njihove teorijske osnove i ključnih aspekata implementacije. Rad je organizovan tako da prvo pruža pregled osnovnih pojmova i tehnika, zatim detaljno opisuje implementaciju algoritama, i na kraju sumira ključne zaključke. Na ovaj način, rad doprinosi boljem razumevanju stohastičkih rešavača i njihovih primena.

2 Osnove

U ovom poglavlju predstavljene su osnovne definicije i notacija koja se koristi u radu. Prvo se uvode osnovni pojmovi formula iskazne logike, kao što su atom, literal i valuacija, zatim se definiše problem zadovoljivosti (SAT problem), a na kraju se opisuje notacija koja će biti korišćena u daljoj analizi algoritama GSAT i WalkSAT.

2.1 Logičke formule

Osnovni gradivni elementi logičkih formula su *atomi*. Atom je promenljiva koja može imati jednu od dve istinitosne vrednosti: *tačno* (`True`) ili *netačno* (`False`).

Literal je atom ili njegova negacija. Na primer, x_1 je pozitivan literal, dok je $\neg x_1$ negativan literal. Literali se povezuju logičkim operatorima, kao što su:

- $\neg$ (negacija),
- $\wedge$ (konjunkcija – "i"),
- $\vee$ (disjunkcija – "ili").

Logička formula se gradi povezivanjem literala pomoću logičkih operatora. Na primer, formula

$$(x_1 \vee \neg x_2) \wedge (x_3 \vee x_4)$$

sastoji se od dve disjunktivne klauze: $(x_1 \vee \neg x_2)$ i $(x_3 \vee x_4)$. Formula je u *konjunktivnoj normalnoj formi* (CNF) kada je zapisana kao konjunkcija klauza, gde je svaka klauza disjunkcija literala.

2.2 Zadovoljivost i SAT problem

Valuacija je dodela istinitosnih vrednosti atomima formule. Na primer, za formulu

$$(x_1 \vee \neg x_2) \wedge (x_3 \vee x_4),$$

valuacija $x_1 = \texttt{True}, x_2 = \texttt{False}, x_3 = \texttt{True}, x_4 = \texttt{False}$ čini formulu tačnom.

Formula iskazne logike se naziva *zadovoljiva* ako postoji valuacija koja je čini tačnom. *Problem zadovoljenja logičkih formula (SAT problem)* postavlja pitanje: *da li postoji valuacija koja čini formulu zadovoljivom?*

2.3 Notacija

U ovom radu formule su predstavljene u CNF obliku. Biće korišćena sledeća notacija:

- x_i: atom formule.
- $\neg x_i$: negacija atoma x_i, tj. negativan literal.
- C_j: klauza (disjunkcija literala, npr. $x_1 \vee \neg x_2$).
- F: formula kao skup klauza, npr. $F = \{C_1, C_2, \ldots, C_m\}$.

Algoritmi GSAT i WalkSAT operišu nad CNF formulama i koriste heuristički pristup za modifikaciju valuacija kako bi pronašli zadovoljivu valuaciju (ukoliko ona postoji).

3 Opis metode

Algoritam koji je implementiran zasniva se na stohastičkim pristupima rešavanja problema zadovoljivosti iskaznih formula. Ovaj tip algoritma koristi lokalnu pretragu prostora rešenja sa ciljem pronalaženja evaluacije koja zadovoljava sve klauze formule. U osnovi, algoritam se oslanja na pretragu sa slučajnim promenama vrednosti literala i iterativno poboljšanje trenutne evaluacije. Pseudokod je prikazan na narednoj slici.

```
procedure StochasticLocalSearch for SAT
    input CNF formula Φ, maxTries, maxSteps
    output satisfying assignment of Φ or "no solution found"
    for i := 1 to maxTries do
        s := initAssign(Φ);
        for j := 1 to maxSteps do
            if s satisfies Φ then return s;
            else
                x := chooseVariable(Φ, s);
                s := s with truth value of x flipped;
            end if
        end for
    end for
    return "no solution found";
end StochasticLocalSearch for SAT
```

Slika 1: Pseudokod stohastičkog SAT rešavača [1]

Na početku izvođenja, algoritam generiše nasumičnu početnu evaluaciju, koja dodeljuje vrednosti promenljivama formule. Zatim se kroz niz iteracija izvršava lokalna pretraga u okviru prostora mogućih rešenja, gde se u svakoj iteraciji bira promenljiva koja će biti promenjena kako bi se što više smanjio broj nezadovoljenih klauzula.

Algoritam koristi dva ključna pristupa za izbor promenljivih:

1. **Odabir literala na osnovu određene heuristike**: Ovaj pristup bazira se na odabiru literala prema određenoj heuristici koja može da uključi kriterijume kao što su broj novih zadovoljenih ili nezadovoljnih klauzula. Na primer, heuristika može izabrati literal koji u trenutnoj evaluaciji uzrokuje najveći broj novih zadovoljenih klauzula, ili koji namanje utice na smanjenje već zadovoljenih klauza.

2. **Nasumičan odabir**: U ovom pristupu, izbor literala se vrši nasumično, uz određenu verovatnoću koja je unapred definisana. Ovaj pristup omogućava algoritmu da istraži širi prostor rešenja, čime se povećava šansa za izlazak iz lokalnih minimuma i pronalaženje boljih rešenja koja možda nisu odmah uočljiva koristeći samo deterministčke strategije.

3.1 GSAT

GSAT algoritam koristi *lokalnu pretragu* sa ciljem da pronađe adekvatnu valuaciju koja zadovoljava sve klauze formule. Algoritam počinje nasumičnom inicijalizacijom valuacije, a zatim iterativno bira i menja vrednost literala, pri čemu se procenjuje ukupan uticaj promena na broj zadovoljenih i nezadovoljenih klauzula. Heuristika koja se koristi u GSAT-u zasniva se na odabiru literala čije promena rezultira maksimalnim povećanjem broja klauza koje postaju zadovoljene, s obzirom na to da promene mogu imati i efekat na nezadovoljene klauzule. Dakle, algoritam ne gleda samo broj klauzula koje će biti zadovoljene nakon promene, već i broj klauzula koje bi mogle postati nezadovoljene.

3.2 WalkSAT

WalkSAT predstavlja naprednu verziju GSAT-a, jer koristi sličnu lokalnu pretragu, ali uz dodatak nasumičnosti. Ovaj pristup omogućava veću fleksibilnost i smanjuje verovatnoću da algoritam zaluta u lokalne minimume, što ga čini efikasnijim u rešavanju teških slučajeva.

U WalkSAT-u, odabir literala može biti zasnovan na dve osnovne strategije:

- **Deterministička odluka:** Odabir literala koji doprinosi najvećem broju zadovoljenih klauzula.
- **Nasumična odluka:** Sa određenom verovatnoćom, odabir nasumičnog literala iz nazadovoljene klauze, čime se omogućava veća fleksibilnost u pretrazi.

4 Implementacija

U ovoj sekciji detaljno ću opisati implementaciju stohastičkog SAT rešavača, uključujući korišćene tehnologije, organizaciju koda, ključne funkcije i biblioteke.

4.1 Jezik implementacije i organizacija koda

Metoda je implementirana u programskom jeziku C++. Struktura koda je organizovana u nekoliko glavnih funkcionalnih delova, uključujući čitanje ulaza, inicijalizaciju podataka, izbor literala, promena vrednosti atoma i proveru zadovoljenosti.

Ključne komponente implementacije su:

- `StochasticSAT` klasa koja sadrži glavnu logiku rešavanja SAT problema, kao i podatke o formuli i trenutnom stanju.
- Funkcija `readDIMACS` koja učitava CNF format sa standardnog ulaza i konvertuje ga u internu reprezentaciju.
- Različite strategije za odabir literala i ažuriranje stanja (`chooseVarBreak`, `chooseVarGain`, `chooseRandomVar`).
- Funkcija `flipAtom` koja menja stanje određenog literala i ažurira unutrašnje stanje rešavača.

4.2 Ključne funkcije i njihove uloge

- `initialize`: Inicijalizuje početno stanje vrednosti literala i broji zadovoljene klauzule. Takodje izračunava početne vrednosti određenih struktura podataka koje su neophodne kako bi algoritam radio što efikasnije.
- `chooseVarBreak`, `chooseVarGain`, `chooseRandomVar`: Implementiraju različite strategije za odabir literala koji će biti promenjeni, zavisno od izabranog pristupa.
- `flipAtom`: Izvršava promenu vrednosti određenog literala i ažurira interno stanje rešavača.
- `isSatisfied`: Na vrlo jednostavan način proverava da li je formula zadovoljena u tekućoj valuaciji.
- `solve`: Glavna funkcija koja pokreće iteracije rešavanja, pokušavajući da pronađe zadovoljivost formule u zadatom broju pokušaja i koraka.

4.3 Strukture podataka gain i breakCount

U implementaciji Stochastic SAT klase se koriste dve ključne strukture podataka: `gain` i `breakCount`. Svaka od njih ima specifičnu ulogu u procesu rešavanja problema, u zavisnosti od toga da li se koristi metoda GSAT ili WalkSAT.

4.3.1 gain

Struktura podataka `gain`, predstavljena u obliku vektora celih brojeva, koristi se u metodi GSAT, koja ima za cilj da poveća broj zadovoljenih klauzula. Elementi u ovom vektoru predstavljaju kumulativni broj klauza koje bi postale zadovoljene odnosno nezadovoljene ako bi se promenila vrednost odgovarajućeg literala.

Uloga: Za svaki literal (atom) izračunava se "dobit" (`gain`), koja predstavlja razliku između broja klauza koje bi postale zadovoljene i broja klauza koje bi postale nezadovoljene u slučaju promene vrednosti datog literala.

Kako se koristi: U izboru literala za promenu vrednosti, algoritam bira literale sa maksimalnim `gain`, jer to znači da će promena tog literala verovatno imati najpozitivniji efekat na broj zadovoljenih klauzula. Ovo je ključni deo za GSAT jer omogućava "pametno"(ali pohlepno) biranje literala koji će doneti najveću korist u rešavanju problema.

Primer: Vrednost 0 u ovoj strukturi ne mora da znači da ne utiče pozitivno ni na jednu klauzu već da je broj klauza koje će ta promena prebaciti iz nezadovoljne u zadovoljnu izjednačen sa brojem klauza koje bi prebacila iz nezadovoljne u zadovoljnu.

4.3.2 breakCount

Ova struktura podataka koja je predstavljena kao vektor celih brojeva, koristi se u metodi WalkSAT, koja ima za cilj da minimizuje broj novonastalih nezadovoljenih klauzula. Elementi u ovom vektoru predstavljaju broj klauzula koje bi postale nezadovoljene ako bi se promenila vrednost odgovarajućeg literala.

Uloga: U WalkSAT metodi, ideja jeste da se minimizuje broj nezadovoljenih klauza, a ne da se maksimizuje broj zadovoljenih. Stoga, za svaki literal se prati broj klauzula u kojima bi se promena tog literala rezultirala nezadovoljstvom. Ako bi promena vrednosti literala x povećala broj nezadovoljenih klauzula, njegov `breakCount` bi bio veći od nula.

Kako se koristi: Algoritam koristi `breakCount` za izbor literala koji minimizuje broj nezadovoljenih klauzula. Literali koji izazivaju najmanje nezadovoljstva su preferirani, dok oni koji bi izazvali veće nezadovoljstvo (tj. povećali broj nezadovoljenih klauzula) nisu poželjni.

Primer: Najpogodnije bi bilo naći neki literal koji u ovoj strukturi ima zapisanu vrednost nula, jer promenom bas njega sigurno neće naškoditi šansi da se dođe do tačnog rešenja.

4.4 Upustvo za prevodjenje i pokretanje

Da biste prevodili i pokrenuli projekat, pratite sledeće korake:

1. Preuzmite i instalirajte C++ prevodilac, kao što je `g++`.
2. Kompajlirajte kod koristeći sledeću komandu:

   ```
   g++ -o StochasticSAT main.cpp
   ```

3. Pokrenite program koristeći sledeću komandu:

   ```
   ./StochasticSAT < input.cnf
   ```

   Gde `input.cnf` predstavlja ulaznu DIMACS datoteku u CNF formatu.

4.5 Potrebni alati i biblioteke

Za prevodjenje i pokretanje programa potrebni su sledeći alati:

- C++ prevodilac (`g++` ili kompatibilni).
- Standardna C++ biblioteka (nije potrebna dodatna biblioteka).

5 Zaključak

Zaključak ovog rada jeste da su GSAT i WalkSAT efikasni heuristički pristupi za rešavanje problema zadovoljivosti formula iskazne logike, međutim, nisu uvek garantovani za pronalaženje rešenja. Zbog njihove heurističke prirode, moguće je da ovi algoritmi ne pronađu rešenje čak i za zadovoljavajuće formule. Ovo je inherentno ograničenje svih heurističkih metoda i predstavlja izazov u primeni na kompleksnije ili veće instance problema.

Potencijalne optimizacije uključuju kombinovanje tehnika koje koriste GSAT i WalkSAT, čime bi se iskoristile prednosti obe metode i mogla povećati ukupna efikasnost rešavanja. Takođe, za budući rad, moguće je implementirati nove metaheurističke pristupe, kao što su genetski algoritam, tabu pretraga i optimizacija kolonijom mrava. Ove metode bi mogle doneti nove načine pretrage i poboljšanja efikasnosti u rešavanju SAT problema.

Iako ovaj rad pruža osnovnu implementaciju oba algoritma, dalja istraživanja i proširenja u ovom pravcu mogli bi značajno unaprediti performanse u rešavanju složenijih slučajeva problema zadovoljivosti.

Literatura

[1] Holger H. Hoos and Thomas Stützle: Local Search Algorithms for SAT: An Empirical Evaluation. (Journal of Automated Reasoning, Vol.24, pp. 421-481, 2000).