Problem SAT i njegovo rješavanje

Irhad Šarić
Prirodno-matematički fakultet u Sarajevu
Kopmjuterska grafika
irhad.saric@hotmail.com

***Abstract* – U ovom radu će biti opisani algoritmi simultanog kaljenja, tabu pretrage i lokalne pretrage za rješavanje problema SAT i njihovi rezultati.**

I. UVOD

SAT problem ili problem zadovoljivosti se sastoji u utvrđivanju da li se promjenljivima nekog bulovskog izraza mogu dodijeliti vrijednosti tako da cijela formula ima vrijednost TAČNO, ili formula ima vrijdnost NETAČNO za sve moguće kombinacije vrijednosti promjenljivih. U drugom slučaju se kaže da je funkcija nezadovoljiva; a u suprotnom je zadovoljiva. Promjenljive su bulovske (imaju vrijednosti 0 ili 1), što znači da je problem binarne prirode.

SAT problem je problem odlučivanja, koji se sastoji iz bulovskih izraza zapisanih samo pomoću operatora I, ILI, NE, promenljivih, i zagrada. U nastavku će se pretpostavljati da je formula zadana u konjuktivnoj normalnoj formi. Formula je u konjuktivnoj normalnoj formi ako predstavlja konjunkciju klauza, gdje je klauza disjunkcija literala.

Pitanje je: ako je dat izraz, da li postoji neka dodjela vrijednosti TAČNO i NETAČNO promjenljivima, takva da cio izraz ima vrijednost TAČNO? Ako je tako, kaže se da je formula zadovoljiva. SAT problem pripada klasi NP-kompletnih problema. Ovaj problem je od ključne važnosti za razne oblasti računarstva, uključujući teorijsko računarstvo, algoritmiku, vještačku inteligenciju i dizajn hardvera.

II. LOKALNA PRETRAGA

Lokalna pretraga pripada grupi S-metaheuristika, koje se zasnivaju na poboljšavanju vrijednosti jednog rešenja. Na početku algoritma se proizvoljno ili na neki drugi način generiše početno rješenje i izračuna vrijednost njegove funkcije cilja. Vrijednost najboljeg rješenja se najprije inicijalizira na vrijednost početnog. Zatim se algoritam ponavlja kroz nekoliko iteracija. U svakom koraku se razmatra rešenje u okolini trenutnog. Ukoliko je vrijednost njegove funkcije cilja bolja od vrijednosti funkcije cilja trenutnog rešenja, ažurira se trenutno rešenje. Takođe se, po potrebi, ažurira i vrijednost najboljeg dostignutog rješenja. Algoritam se ponavlja dok nije ispunjen kriterijum zaustavljanja. Kriterijum zaustavljanja može biti, na primer, dostignut maksimalan broj iteracija, dostignut maksimalan broj ponavljanja najboljeg rješenja, ukupno vrijeme izvršavanja, itd. Lokalna pretraga se može prikazati sljedećim pseudokodom:

```
1. Generisati početno rješenje s
2. Inicijalizirati vrijednost najboljeg
   rješenja (broj zadovoljenih klauza)
3. While poboljšanje nađeno do:
      a. Uzeti svako rješenje iz
         susjedstva
      b. Izračunati da li ono daje bolji
         rezultat od najboljeg i za
         koliko
      c. Zapamtiti susjeda koji daje
         najbolji rezultat i početi novu
         iteraciju od njeg.
```

III. TABU SEARCH

Osnovna ideja ovog tipa heurističkog algoritma jest zamijeniti dopustivo rješenje *X* novim, najboljim rješenjem *Y* iz njegove okoline, koje je također dopustivo. Treba biti pažljiv jer moglo bi se dogoditi da u sljedećem koraku zamijenimo *Y* s *X* pa bismo se počeli vrtjeti u krug, što nikako ne želimo. Zato se izgrađuje "*tabu-lista*" u kojoj se nalaze neka rješenja koja algoritam ne smije odabrati u sljedećem koraku. U tabu-listi ne nalaze se sva dopustiva rješenja koja je algoritam u nekom koraku posjetio, već samo ona posjećena u zadnjih *L* iteracija (uobičajeno je $L = 10$). Za potpuno onemogućavanje ciklusa, *L* bi trebao biti jako velik, no tada ne bi bilo mnogo dozvoljenih odabira i bilo bi ih teško naći. Time bi se program usporio, što je u suprotnosti s onim što se heurističkim algoritmima nastoji postići: dopušta se lošiji rezultat, ali se omogućava veća brzina izvođenja programa.

Za problem SAT, kada se neka varijabla u rješenju promijeni, u tabu listu se na određeni index upisuje broj poteza za koje ova varijabla neće biti dostupna za mijenjanje. Pseudokod je dat u nastavku:

```
 1 sBest ← s0
 2 bestCandidate ← s0
 3 tabuList ← []
 4 tabuList[0] ← forbiddenMoves
 5 while (not stoppingCondition())
 6 sNeighborhood ←
getNeighbors(bestCandidate)
 7      for (sCandidate in
sNeighborhood)
 8              if ( (not
tabuList.contains(sCandidate)) and
(fitness(sCandidate) >
fitness(bestCandidate)) )
 9                      bestCandidate ←
sCandidate
10              end
11      end
12      if (fitness(bestCandidate) >
fitness(sBest))
13              sBest ← bestCandidate
14      end
15      tabuList.push(bestCandidate)
16      decreaseAllPositiveNumbers
(tabuList)
19 end
20 return sBest
```

IV. SIMULTANO KALJENJE

Algoritam simuliranog kaljenja, kao i tabu algoritam, omogućava bijeg iz lokalno optimalnog rješenja (koje ne mora nužno biti i globalno optimalno), a bazira se na metodi oplemenjivanja metala koja se naziva kaljenjem i koristi se već 7000 godina. Proces se sastoji od 3 faze:

zagrijavanja na visoku temperaturu, zadržavanja na toj temperaturi i zatim hlađenja, obično na sobnu temperaturu. Polaganim hlađenjem inducira se promjena kristalne rešetke metala tako da njezina energija postaje najmanja moguća. Tako metali postaju čvršći. Hlađenje je potrebno obaviti vrlo pažljivo: ako se ohladi prebrzo, može doći do pucanja metala, a potrebno je i neko vrijeme kako bi došlo do potrebnih transformacija u molekularnoj strukturi metala.

U općenitom slučaju naša je želja maksimalizirati neku funkciju P, tj. pronaći takav X da je $P(X)$ maksimalna. Ova metoda aktualno dopustivo rješenje X zamjenjuje nekim njemu bliskim dopustivim rješenjem Y ako je

$$P(Y) \geq P(X)$$

No, ponekad je i u slučaju kada je $P(Y) < P(X)$ dopušteno X zamijeniti s Y: ako za slučajno generirani $r \in [0,1]$ vrijedi:

$$r < e^{\frac{P(Y)-P(X)}{T}}$$

X će se zamijeniti s Y. Na taj način algoritam neće zaglaviti u lokalnom minimumu.

Kako ova metoda simulira hlađenje, kontrolna varijabla koja oponaša padanje temperature (T) iznimno je važna, a njezina je početna vrijednost $T_0 > 0$. Tijekom izvođenja programa varijabla T smanjuje se prema određenom rasporedu hlađenja. U početku će vjerojatnost zamjene biti velika pa će biti više zamjena X i Y, no sa svakom iteracijom ta će vjerojatnost padati i zamjene koje smanjuju vrijednost funkcije P postajat će rjeđe. U nastavku je dat pseudokod za simultano kaljenje.

```
procedure simulated annealing
begin
  t ← 0
  initialize T
  select a current point v_c at random
  evaluate v_c
  repeat
    repeat
      select a new point v_n
        in the neighborhood of v_c
      if eval(v_c) < eval(v_n)
        then v_c ← v_n
        else if random[0, 1) < e^((eval(v_n)-eval(v_c))/T)
            then v_c ← v_n
    until (termination-condition)
    T ← g(T, t)
    t ← t + 1
  until (halting-criterion)
end
```

Bitno je napomenuti da se za rješavanje ovog problema temperature računa po formuli $T = T_{MAX} * e^{-t*r}$ gdje je t broj ponavljanja algoritma a r koeficijent hlađenja, a on se računa po formuli: $\frac{1}{t*broj\_varijabli}$

V. REZULTATI

Algoritmi su pokrenuti samo jednom nad istim početnim slučajnim rješenjem. Rezultati su prikazani u tabeli ispod:

Broj varijabli	Broj klauza	Lokalna pretraga	Tabu	Simultano
20	91	88	90	91
50	218	211	214	217
75	325	314	319	325
100	430	423	425	429
125	538	524	531	538
150	645	630	640	645
175	753	738	746	750
200	860	836	855	858
225	960	941	951	957
250	1065	1037	1059	1065

VI. REFERENCE

[1] How to Solve It – Modern Heuristics (Zbigniew Michalewicz, David B. Fogel)

[2] https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

[3] https://en.wikipedia.org/wiki/Tabu_search#Pseudocode

[4] http://e.math.hr/old/heuristicki/index.html#4