Rešavanje problema maksimalne k-zadovoljivosti optimizacijom rojem čestica

David Dimić
Zorana Gajić

Mart 2019.

Sadržaj

1 Uvod

2 Formulacija problema

Data je formula F u KNF obliku sa n promenjivih $(x_1, x_2, ..., x_n)$ i m klauza. Problem maksimalne k-zadovoljivosti može se definisati na sledeći način:

Klauza C_i dužine k je disjunkcija k literala: $C_i = (x_1 \vee x_2 ... \vee x_k)$, gde je svaki literal promenjiva ili njegova negacija i može se pojavljivati više puta u izrazu. Cilj je pronaći istinitosne vrednosti promenjivih, valuaciju koja je vektor $\vec{v} = (x_1, x_2, ..., x_n) \in \{0, 1\}^n$ tako da ova valuacija maksimizuje broj zadovoljenih klauza u formuli F.

Ako valuacija zadovoljava formulu, onda se ona naziva modelom formule. Max k-SAT problem može se definisati parom (Ω, SC), gde je Ω skup svih potencijalnih rešenja iz $\{0, 1\}^n$, a $SC : \Omega \to N$, skor valuacije koji je jednak broju zadovoljenih klauza. Shodno ovome, problem max k-SAT je naći $\vec{v} \in \Omega$ za koje je SC maksimalno:

$$\max_{\vec{v} \in \Omega} \{SC(\vec{v})\}$$

Očigledno ima 2^n potencijalnih rešenja koji zadovoljavaju formulu F. Problem max k-SAT je NP-kompletan za svako $k > 2$.

3 Optimizacija rojem čestica (PSO)

Optimizacija rojem čestica (Particle swarm optimization – PSO) je jedna od tehnika pretraživanja zasnovana na populaciji kao što je genetski algoritam, ali ne koriste evolutivne algoritme kao što su mutacija i ukrštanje. PSO algoritmi su 1995. godine uveli Kenedi i Eberhart kao alternativu standardnim genetskim algoritmima.

Optimizacija rojem čestica je algoritam zasnovan na ponašanju pojedinačnih jedinki unutar određene grupe (na primer, jata ptica ili roja insekata). Ukoliko se, vođeno instiktom, jato prica uputi u određenom smeru u potrazi za hranom, očekivanje je da će čitavo jato slediti upravo onu pticu koja je pronašla izvor hrane. Međutim, i svaka ptica ponaosob može biti vođena sopstvenim instiktom i time na trenutak u potrazi za hranom napustiti jato. Tada se verovatno može desiti da, ukoliko pronađe bolji izvor hrane, čitavo jato upravo krene da sledi tu pticu.

PSO pripada skupu algoritama koji se zasnivaju na inteligenciji roja (swarm intelligence). Algoritam radi nad skupom jedinki, koji se naziva rojem. Elementi ovog skupa se nazivaju česticama. Svaka čestica predstavlja kandidatsko rešenje optimizacionog problema. Čestice se na unapred definisan način kreću po prostoru pretraživanja. Njihovo kretanje se usmerava imajući u vidu njihovu trenutnu poziciju, njihovu do sada najbolju poziciju, kao i do sada najbolju poziciju čitavog roja. Pod najboljom pozicijom čitavog roja se podrazumeva do sada najbolja pozicija, uzimajući u obzir sva njegova rešenja. Proces se ponavlja dok ne bude zadovoljen kriterijum zaustavljanja, a u svakoj iteraciji se ažurira najbolja vrednost rešenja za svaku česticu, kao i za roj u celini.

Neka je dat roj sa $\vec{S}$ čestica. Svaka čestica se sastoji od tri elementa:

- Pozicija u prostoru za pretragu $\vec{x_i}$
- Brzina, vektor $\vec{v_i}$
- Sećanje, koje se koristi za skladištenje elitnih čestica globalne pretrage $\vec{P_g}$, kao i najboljih individualnih rešenja $\vec{P_i}$ koja su do sada pronašle zasebne čestice

Nije neophodno da se u budućim populacijama nalazi bilo koji elitni pojedinac, iako svaka čestica u populaciji pokušava da bude blizu svog najboljeg rešenja i globalnog najboljeg rešenja.

Osnovni oblik PSO algoritma dat je sledećim samoažurirajućim jednačinama:

$$\vec{v_i}^{t+1} = w \cdot \vec{v_i}^{t} + c_1 \cdot \vec{r_1} \times (\vec{P_i}^{t} - \vec{x_i}^{t}) + c_2 \cdot \vec{r_2} \times (\vec{P_g}^{t} - \vec{x_i}^{t}) \tag{1}$$

$$\vec{x_i}^{t+1} = \vec{x_i}^{t} + \vec{v_i}^{t+1} \tag{2}$$

Jednačina 1 opisuje kako se ažurira brzina i-te čestice, a 2 koja je sledeća pozicija i-te čestice, pri čemu je:

- w - faktor inercije
- c_1, c_2 - faktori učenja: kognitivna i socijalna
- $\vec{v_{id}}^{t}$ - brzina i-te čestice u iteraciji t
- $\vec{x_{id}}^{t}$ - pozicija i-te čestice u iteraciji t
- $\vec{r_1}, \vec{r_2}$ - pseudoslučajni brojevi iz uniformnog intervala $[0, 1]$
- $\vec{P_i}$ - najbolje individualno rešenje čestice i
- $\vec{P_g}$ - trenutno najbolje globalno rešenje

Kako je max k-SAT problem diskretan potrebno je prilagoditi jednačinu 2. Izračunata brzina $\vec{v_i}$ je iz $\mathbb{R}^n$, pa je potrebno da se svede na $\{0, 1\}^n$. Jedan predlog za ažuriranje položaja čestice, izložen u radu [1], dat je sigmoidnom transformacijom. Sada ${v_i}^{t}$ predstavlja verovatnoću da bit x_i^t uzme vrednost 1.

$$x_i^t = \begin{cases} 1, rand(0,1) < sigmoid(v_i^t) \\ 0, inace \end{cases} \tag{3}$$

$$sigmoid(v_i^t) = \frac{1}{1 + e^{-v_i^t}} \tag{4}$$

3.1 Pseudokod PSO

Input : Formula F u KNF-u, n i m
Output: Najbolja procenjena valuacija i broj zadovoljenih klauza

inicijalizacija populacije: pozicije i brzine;
t = 0; `// tekuca iteracija`
while *nije zadovoljen uslov zaustavljanja* **do**
    t = t + 1;
    for $i \leftarrow 0$ **to** *broj cestica u roju* **do**
        Izračunaj $Fitness(\vec{P_i^t})$;
        Sačuvaj individualni najbolji rezultat kao globalni $\vec{P_g}$;
        Ažuriraj brzine na osnovu $\vec{P_i}$ i $\vec{P_g}$;
        Ažuriraj pozicije $\vec{v_i^t}$;
        Ažuriraj individualni najbolji rezultat $\vec{P_i}$;
        Ažuriraj globalni najbolji rezultat $\vec{P_g}$;
    end
end

Algorithm 1: Uopšteni PSO algoritam

3.2 Reprezentacija rešenja

Za dobar algoritam je takođe važno dobro predstavljanje rešenja. Postoje više načina reprezentacija, ali je ovde odabran prirodna binarna reprezentacija. Svaka čestica je predstavljena binarnim nizom dužine n.

3.3 Inicijalizacija

Potrebno je inicijalizovati pozicije čestica i vektora brzine. Pozicije su inicijalizovane pseudo-slučajnim brojevima $\{0, 1\}$, a vektor brzine realnim brojevima iz intervala $[-V_{min}, V_{max}]$, gde su granice intervala jedan od parametara PSO algoritma.

3.4 Fitnes funkcija

Fitnes funkcija je veoma važna za perfomanse algoritma. Prva fitnes funkcija koja se sama nameće jeste broj zadovoljenih klauza. Ova funkcija se nije pokazala kao dovoljno dobra, a bolji mehanizam je stepenasto ažuriranje težina (SAW - stepwise adaptation weights) uvedena od strane Eiben-a [2]. Fitnes funkcija je data sledećom formulama:

$$F_{SAW}(x) = \sum_{i=1}^{m} W_i C_i(x) \tag{5}$$

$$W_{i+1} = W_i + 1 - C_i(x^*) \tag{6}$$

Svakoj klauzi C_i dodeljuje se težina W_i. Ova funkcija ima za cilj identifikovanje težih klauza u procesu učenja koja je predstavljena većom vrednošću W_i. Na početku su težine inicijalizovane na 1, pa se potom ažuriraju jednačinom 6. x^* je tekuće najbolje rešenje.

3.5 Lokalna pretraga i flip heuristika

Da bi se unapredio standardni PSO algoritam uvedena je, umesto ažuriranja pozicija jednačinom 3, stohastička lokalna pretraga flip heuristikom.

Input : pozicija čestice p, Formula F u KNF-u, maxFlip
Output: nova pozicija čestice p

```
improvement = 1;
numFlip = 0;
while improvement > 0 and numFlip < maxFlip do
    improvement = 0;
    for i ← 0 to n do
        flip p[i];
        numFlip += 1;
        Izračunaj dobit: gain;
        if gain >= 0 then
            prihvati flip;
        end
        else
            odbaci flip, vrati na staru vrednost p[i];
        end
    end
end
```

Algorithm 2: Funkcija lokalne pretrage

Parametar	Vrednost
Broj iteracija	1000
w	1
c1	1.7
c2	2.1
Broj čestica	20
max flip	30000
v_{min}	-1
v_{max}	1

Tabela 1: Parametri

3.6 Kriterijum zaustavljanja

Jedini kriterijumi zaustavljanja koji se mogu koristiti u opštem slučaju su: da li je dostignut maksimalan broj unapred zadatih iteracija ili, da li u poslednjih nekoliko iteracija nema značajnog napretka. Ali, u test primerima za koje unapred znamo da je formula zadovoljiva, ili koliko je klauza maksimalno zadovoljivo, možemo koristiti i kriterijum da je broj zadovoljenih klauza dostigao ukupan broj klauza m, odnosno $m - l$, gde je l broj klauza koji se ne mogu zadovoljiti.

3.7 Varijante PSO algoritma

Da bismo uporedili kombinaciju lokalne pretrage, SAW fitnes funkcije i klasičnog PSO algoritma implementirani su i testirane sledeće tri verzije:

PSO-LS - Osnovna varijanta algoritma koji ne koristi lokalnu pretragu, već sigmoidnu transformaciju, jednačine 1 i 3 za ažuriranje brzina i pozicije čestica. Fitnes funkcija je F_{SAW}, sa korišćenjem težina nad klauzama. Karakteriše ga sporija konvergencija do globalnog optimuma, ali pojedinačne iteracije se izvršavaju brže.

PSOSAT - Koristi lokalnu pretragu, ali ne i F_{SAW}, pa je funkcija cilja broj zadovoljenih klauza. Mana ovog algoritma je teško izlaženje iz lokalnih optimuma zbog korišćenja fitnes funkcije koja ne raznanaje težinu klauza.

WPSOSAT - Modifikovan PSO algoritam sa korišćenjem flip heuristike i F_{SAW} fitnes funkcije. Značaj lokalne pretrage ogleda se u poređenju sa PSO-LS, a korišćenje F_{SAW} u poređenju sa PSOSAT.

3.8 Rezultati

Svi algoritmi pokretani su pet puta sa istim parametrima i beležen je prosečan broj zadovoljenih klauza, pri čemu podebljan rezultat označava da se do rešenja dolazilo u prvoj iteraciji. U tabeli 2 skoro svi algoritmi su uspeli da brzo nađu rešenje. Za sada između PSOSAT i WPSOSAT nema razlike. Već za poslednji test vidi se slabost ne korišćenja lokalne pretrage. Već u tabeli 3 PSO-LS nije mogao da se uporedi sa ostala dva algoritma. Za neke instance do globalnog optimuma došao je jedino WPSOSAT odakle se vidi značaj SAW funkcije.

Instanca	Broj literala	Broj klauza	PSO-LS	PSOSAT	WPSOSAT
aim-50-1_6-no	50	80	79	**79**	**79**
aim-50-2_0-no	50	100	99	**99**	**99**
aim-100-1_6-no	100	160	159	**159**	**159**
aim-100-2_0-no	100	200	199	**199**	**199**
aim-200-2_0-yes	200	400	398.6	**399**	**399**

Tabela 2: AIM nezadovoljivi testovi

Instanca	Broj literala	Broj klauza	PSO-LS	PSOSAT	WPSOSAT
aim-50-1_6-yes	50	80	79	79.2	80
aim-50-2_0-yes	50	100	99	99	100
aim-50-3_4-yes	50	170	168.6	170	170
aim-50-6_0-yes	50	300	300	300	300
aim-100-1_6-yes	100	160	158.6	159	159
aim-100-2_0-yes	100	200	198	199	200
aim-100-3_4-yes	100	340	328	339.4	340
aim-100-6_0-yes	100	600	580.2	**600**	**600**
aim-200-2_0-yes	200	400	395.4	399	399
aim-200-6_0-yes	200	1200	1135.6	**1200**	**1200**

Tabela 3: AIM zadovoljivi testovi

Literatura

[1] Kennedy J, Eberhart RC. "A discrete binary version of the particle swarm algorithm," In: Proceedings of IEEE conference on systems, man and cybernetics , p. 4104–4109, 1997.

[2] Gottlieb J, and Voss N. "Adaptive fitness functions for the satisfiability problem". In: Parallel Problem Solving from Nature - PPSN VI 6th International Conference, Paris, France. Springer Verlag. LNCS 1917, 2000.

[3] Abdesslem Layeb, "A particle swarm algorithm for solving the maximum satisfiability problem", MISC Lab., Computer science department, University mentouri of Constantine, Algeria