

Optimizacija maksimalne k-zadovoljivosti populacionim metahauristikama

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

David Dimić, Zorana Gajić
daviddimic@hotmail.com, zokaaa_gajich@bk.ru

Mart 2019.

Sadržaj

1	Uvod	3
2	Formulacija problema i reprezentacija rešenja	3
3	Evolucionni algoritmi (EA)	3
3.1	Pseudokod EA	4
3.2	Inicijalizacija rešenja	4
3.3	Fitnes funkcija	4
3.3.1	Fitnes funkcija SAW	4
3.3.2	Fitnes funkcija RFEA	5
3.4	Selekcija	5
3.4.1	Ruletska selekcija	5
3.4.2	Turnirska selekcija	5
3.5	Ukrštanje	6
3.6	Mutacija	6
3.6.1	Slučajna mutacija	6
3.6.2	Slučajna-prilagodljiva mutacija	6
3.6.3	Mutacija zasnovana na znanju	6
3.6.4	Mutacija jednog bita	6
3.6.5	Lamarckian SEA-SAW mutacija	6
3.7	Politika zamene generacija	6
3.8	Lokalna pretraga	7
3.9	Varijante EA algoritma	7
3.9.1	SAWEA	7
3.9.2	RFEA	7
3.9.3	FlipGA	8
3.9.4	ASAP	8
3.10	Rezultati	8
4	Optimizacija rojem čestica (PSO)	8
4.1	Pseudokod PSO	10
4.2	Inicijalizacija rešenja	10
4.3	Fitnes funkcija	10
4.4	Lokalna pretraga i flip heuristika	11
4.5	Varijante PSO algoritma	11
4.5.1	PSO-LS	11

4.5.2	PSOSAT	11
4.5.3	WPSOSAT	11
4.6	Rezultati	12
5	Zaključak	13
	Literatura	13

1 Uvod

2 Formulacija problema i reprezentacija rešenja

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

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

Ako valuacija zadovoljava formulu, onda se ona naziva modelom formule F . Max k -SAT problem može se definisati parom (Ω, SC) , gde je Ω skup svih potencijalnih rešenja iz $\{0, 1\}^n$, vektor n promenljivih, a $SC : \Omega \rightarrow \mathbb{N}$, 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 . Dokazano je da je problem max k -SAT je NP-kompletan za svako $k > 2$.

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 Evolucionari algoritmi (EA)

Genetski algoritmi [5] su bazirani na Darwinovoj teoriji evolucije, u kojoj unutar jedne populacije najčešće opstaju najbolje prilagođene jedinke. Reprezentacija jedinke naziva se hromozom ili genotip. Cilj je naći vrednost za koju zadata funkcija cilja dostiže svoj ekstremum ili vrednost koja je dovoljno blizu ekstremuma. Potencijalna rešenja, tj. jedinke obično su predstavljene nizovima nula i jedinica, ali moguće su i druge reprezentacije. Postupak se odvija kroz generacije.

U svakoj generaciji postoji isti broj jedinki i za svaku od njih računa se njen kvalitet. Funkcija koja pridružuje te vrednosti jedinkama naziva se funkcija prilagođenosti ili fitness funkcija. Iz jedne generacije se na osnovu vrednosti fitness funkcije, kroz proces selekcije biraju jedinke koje će biti iskorišćene za stvaranje novih jedinki. One kvalitetnije se biraju sa većom verovatnoćom. Videćemo različite tehnike selekcije kao što su turnirska i ruletska.

Nad izabranim jedinkama primenjuju se genetski operatori ukrštanja i tako se dobijaju nove jedinke. Ukrštanjem se od dve jedinke dobija nova (ili dve nove) sa genetskim materijalom koji je dobijen neposredno od roditelja, tj. od polaznih jedinki. U ovom radu smo koristili samo uniformno ukrštanje.

Operatorom mutacije može da se modifikuje deo polazne jedinke. U svakoj generaciji, dakle, može da dođe do rekombinacije gena zbog koje se javlja sličnost ali i različitost između jedinki iste generacije. Videćemo različite vrste mutacija: slučajna, zasnovana na znanju, jednog bita i ostale.

Politika zamene generacija određuje kako se od postojećih jedinki i njihovog potomstva kreira nova generacija. Neke jedinke u novoj generaciji mogu biti bolje, a neke lošije od jedinki u prethodnoj generaciji.

Kriterijumi zaustavljanja mogu biti razni, ali smo koristili zaustavljanje postupka kada se dostigne zadati broj iteracija.

3.1 Pseudokod EA

U ovom poglavlju predstavimo osnovni oblik algoritma na kojem se dalje zasnivaju ostale varijante.

Input : Formula F u KNF-u, n i m
Output: Najbolja procenjena valuacija i broj zadovoljenih klauza
inicijalizacija populacije: pozicije i brzine;
 $t = 0$; // **tekuća iteracija**
while *nije zadovoljen uslov zaustavljanja* **do**
 $t = t + 1$;
 Selekcija jedinki za primenu genetskih operatora;
 Ukrštanje za izabrane parove jedinki;
 Mutacija izabranih jedinki;
 Određivanje funkcije prilagođenosti populacije;
end

Algorithm 1: Osnovni Evolutivni algoritam

3.2 Inicijalizacija rešenja

Populaciju jedinki jedne generacije čini skup nizova binarnih cifara. Potrebno je da inicijalizujemo hromozome pseudoslučajnim brojevima $\{0, 1\}$. Veoma važnu ulogu igra odabir parametara. Parametri mogu biti fiksni ili im se vrednosti mogu na određen način menjati iz generacije u generaciju.

3.3 Fitnes funkcija

Tokom različitih varijanti EA koristile su se različite funkcije prilagođenosti. Prva fitness funkcija koja se sama nameće jeste broj zadovoljenih klauza, koju ćemo koristiti u FlipGA i ASAP algoritmima.

$$f_{MAXSAT}(x) = \sum_{k=1}^N f(x, c_k)$$

gde je N ukupan broj klauza, x predstavlja jedan hromozom i

$$f(x, c_k) = \begin{cases} 1, & \text{ako je klauza } c_k \text{ zadovoljena hromozomom } x \\ 0, & \text{inače.} \end{cases}$$

Nešto malo komplikovanije funkcije prilagođenosti se koriste u SAWEA i RFEA algoritmima.

3.3.1 Fitnes funkcija SAW

Eiben i van der Hauw su razvili evolutivni algoritam koji koristi stepenasto prilagođavanje težina (eng. *stepwise adaptation of weights*), odnosno:

$$w_i = w_i + \Delta w$$

gde w_i predstavlja težinu.

SAW funkcija prilagođenosti će nakon nekog vremena povećati težine nezadovoljenih klauza. Za jedan hromozom i definišemo funkciju prilagođenosti kao:

$$f_{SAW}(i) = w_1 f(i, c_1) + \dots + w_N f(i, c_N) = \sum_{k=1}^N w_k f(i, c_k)$$

gde $w_k \geq 1$ označava težinu.

Inicijalno, težine su postavljene na vrednost 1. Za Δw se koristi formula:

$$\Delta w = 1 - f(x^*, c_i)$$

gde je x^* trenutno najbolji hromozom u populaciji.

Težina w_i klauze c_i se povećava za 1 ako trenutna najbolja jedinka x^* ne zadovoljava klauzu c_i , dok Δw povećava težine nezadovoljenih klauza. Visoke težine su dobijene klauzama koje nisu zadovoljene duže vreme.

3.3.2 Fitnes funkcija RFEA

Gotlieb i Vos su predstavili koncept funkcija prerada (eng. *refining functions*) koje su nastale tako što smo želeli da prevaziđemo manu obične funkcije prilagođenosti za naš problem koja vrlo često vraća isti rezultat za različite hromosome. Definišemo je koristeći f_{MAXSAT} za hromozom x na sledeći način:

$$f_{REF}(x) = f_{MAXSAT}(x) + \alpha r(x)$$

gde je $\alpha > 0$ predstavlja nivo uticanja funkcije r , $r : \{0, 1\}^n \rightarrow [0, 1]$. Za $\alpha = 0$ f_{REF} se ponaša kao f_{MAXSAT} .

Funkcija r za hromozom x se definiše na sledeći način:

$$r(x) = \frac{1}{2} \left(1 + \frac{\sum_{i=1}^N K(x_i) v_i}{1 + \sum_{i=1}^N |v_i|} \right)$$

gde $K(0) = -1, K(1) = 1$. Autori su predložili da ažuriranje v_i izgleda ovako:

$$v_i = v_i - K(x^*) \sum_{k \in U_i(x^*)} w_k$$

gde je x^* trenutno najbolja jedinka, $U_i(x^*)$ je skup nezadovoljenih klauza u kojima se pojavljuje promenljiva i , w_k je definisano na isti način kao i u 3.3.1, v_i predstavlja težine promenljivih.

Visoke pozitivne težine ukazuju da se za odgovarajuće promenljive preferira da budu 1, dok za negativne 0. Inicijalno, sve težine promenljivih su postavljene na vrednost 0.

3.4 Selekcija

Selekcija predstavlja izbor jedinki iz trenutne populacije koje će biti korišćene za dobijanje naredne generacije. Obezbeđuje čuvanje i prenošenje dobrih osobina populacije. Koristili smo dve najpopularnije strategije selekcije: ruletska i turnirska.

3.4.1 Ruletska selekcija

Ruletska selekcija (eng. *roulette wheel selection*) [5] je proces selekcije u kojem veće šanse da učestvuju u reprodukciji imaju prilagođenije jedinke.

Ako $f(i)$ vrednost funkcije prilagođenosti za jedinku i , a N broj jedinki u populaciji, verovatnoća da će jedinka i biti izabrana da učestvuje u reprodukciji jednaka je:

$$p_i = \frac{f(i)}{\sum_j^N f(j)}$$

Ruletsku selekciju smo primenili u FlipGA varijanti EA.

3.4.2 Turnirska selekcija

U turnirskoj selekciji [5] jedinke odigravaju turnire u kojima veće šansu za pobedu imaju one sa boljom prilagođenošću.

Za jedan turnir bira se slučajno k jedinki iz populacije. Pobednikom se smatra jedinka sa najvećom prilagođenošću.

Turnirsku selekciju smo primenili u REF varijanti EA.

3.5 Ukrštanje

U ukrštanju [5] učestvuju dve jedinke koje se nazivaju roditelji. Rezultat ukrštanja je jedna nova jedinka ili dve nove jedinke koje nazivamo decom. U samo jednoj varijanti EA, FlipGA, smo koristili ukrštanje, a tačnije uniformno ukrštanje.

Uniformno ukrštanje daje dva deteta. Kod ovog ukrštanja svaki bit prvog roditelja se sa verovatnoćom p prenosi na prvo dete i sa verovatnoćom $1 - p$ na drugo dete. Za verovatnoću p smo uzeli vrednost 0.5.

3.6 Mutacija

Mutacija [5] se primenjuje nakon ukrštanja. To je operator koji sa određenom verovatnoćom menja jedan deo jedinke na određeni način.

Uloga mutacije je da spreči da jedinke u populaciji postanu suviše slične i da pomogne u obnavljanju izgubljenog genetskog materijala.

3.6.1 Slučajna mutacija

Najklasičnija vrsta mutacije je slučajna mutacija (eng. *random mutation*). Na slučajan način biramo vrednost $t \in [0, 1]$ i ako je ta vrednost manja od zadatog parametra mutiranja *mutation_rate* tj. $t < \text{mutation_rate}$ onda vršimo invertovanje na slučajnoj poziciji hromozoma.

Ovakav operator mutiranja koristimo u FlipGA

3.6.2 Slučajna-prilagodljiva mutacija

Ovakav operator mutacije se koristi u ASAP.

3.6.3 Mutacija zasnovana na znanju

Kod mutacije zasnovane na znanju (eng. *knowledge-based mutation*) iz skupa nezadovoljenih klauza biramo tačno jednu klauzu i iz nje dobijamo promenljivu izabranu na slučajan način. Apsolutna vrednost te promenljive predstavlja poziciju invertovanja u hromozomu.

Ovakav operator mutacije se koristi u RFEA.

3.6.4 Mutacija jednog bita

Jedna od najjednostavnijih vrsta mutacije je mutacija jednog bita (eng. *mutation one operator*). Ovakav operator sa verovatnoćom 1 invertuje tačno jedan bit hromozoma na slučajno odabranoj poziciji.

Ovakav operator mutacije se koristi u SAWEA.

3.6.5 Lamarckian SEA-SAW mutacija

Da bismo poboljšali SAWEA i RFEA koristimo Lamarckian mutaciju koju su predstavili de Jong i Kisters. Ovakav operator mutacije liči na lokalnu pretragu. Hromozom c od skupa dece je odabran na slučajan način. Takođe kreiramo skup slučajno odabranih klauza. Ako je svaka klauza u ova-kvom skupu zadovoljena hromozomom c onda ne raditi ništa. Inače, izabрати slučajnu promenljivu u nezadovoljenoj klauzi i invertovati odgovarajući bit tako da klauza bude zadovoljena.

3.7 Politika zamene generacija

Politika zamene generacije [5] opisuje kako se od tekuće generacije dobija nova. Osnovna podela po ovom kriterijumu je na generacijske genetske algoritme (eng. *generational genetic algorithm*) i genetske algoritme stabilnog znanja (eng. *steady state genetic algorithm*).

U slučaju generacijskih genetskih algoritama, nova generacija se dobija tako što se selekcijom bira dovoljno jedinki iz tekuće generacije da se napravi cela nova generacija. Izabrane jedinke se ukrštaju i mutiraju i tako dobijena generacija zamenjuje staru.

U slučaju genetskih algoritama stabilnog stanja, čim se izabere par roditelja, vrši se ukštanje i mutacija i umetanje potomaka u populaciju u skladu sa nekom politikom zamene.

U našem radu smo pored običnog generacijskog genetskog algoritma, koristili još dve vrste: (μ, λ) i $(\mu + \lambda)$, gde μ predstavlja broj pojedinaca u populaciji, a λ broj dece.

(μ, λ) - od μ roditelja kreira λ dece i za novu generaciju se uzimaju najbolji od dece.

$(\mu + \lambda)$ - od μ roditelja kreira λ dece i bira najbolje jedinke iz skupa roditelja i dece zajedno.

3.8 Lokalna pretraga

3.9 Varijante EA algoritma

Da bismo dobili što bolje rešenje pokušavamo da kombinujemo različite tehnike operatora selekcije, ukrštanja, mutacije. U tabeli možemo da vidimo osnovna svojstva varijanti evolutivnog algoritma koje smo implementirali.

Tabela 1: Varijante EA algoritma

Svojstvo	SAWEA	RFEA	FlipGA	ASAP
zamena	$(1, \lambda^*)$	steady-state	generational	$(1 + 1)$
selekcija	-	tournament	roulette	-
fitness	f_{SAW}	f_{REF}	f_{MAXSAT}	f_{MAXSAT}
ukrštanje	-	-	uniform	-
mutacija	mut-one	knowledge-based	random	random-adaptive
lokalna pretraga	-	-	flip heuristic	flip heuristic
adaptacija	fitness	fitness	-	tabu list

Zajedničko za sve vrste EA je inicijalizacija koja se vrši na slučajan način.

3.9.1 SAWEA

SAWEA - evolutivni algoritam koji koristi stepenasto prilagođavanje težina (eng. *Using Stepwise Adaptation of Weights*) [3, 1] koji su razvili Eiben i van der Hauw. Korišćena funkcija prilagođenosti je f_{SAW} koja je objašnjena u 3.3.1. Težine w_i su na početku inicijalizovane na vrednost 1.

Prema preporuci autora koji su osmislili ovaj algoritam u našem radu koristimo politiku zamene generacije $(1, \lambda^*)$ 3.7 i mutaciju jednog bita 3.6.4.

S obzirom da je politika zamene $(1, \lambda^*)$ iz ovoga zaključujemo da nam je populacija veličine 1, zbog čega nije moguće vršiti ukrštanje. Kao ulazni argument ovog algoritma pored ostalih imamo i parametar λ^* koji po default-u uzima vrednost 10. Kreiranje nove generacije izgleda ovako: uzimamo jednog jedinog mogućeg roditelja, λ^* puta primeni operator mutacije da bismo dobili najviše λ^* dece i za novu generaciju uzeti najbolju jedinku iz skupa dece.

Ne zaboraviti nakon ovoga izvršiti ažuriranje težina klauza.

3.9.2 RFEA

Gotlieb i Vos [3, 1] su predstavili koncept funkcija prerada (eng. *refining functions*). Više u korišćenoj funkciji prilagođenosti 3.3.2. U našem algoritmu mi zapravo koristimo SAWEA u kombinaciji sa klasičnim RFEA.

RFEA ima populaciju veličine 4. Koristi turnirsku selekciju 3.4.2 veličine 2 i politiku zamene generacije stabilnog stanja 3.7 sa eliminacijom najgore jedinke. Takođe je korišćena eliminacija duplikata, tj. dete je odbijeno ukoliko je već u populaciji. Jedini variacioni operator je mutacijski operator zasnovan na znanju 3.6.3. Ne zaboraviti nakon ovoga izvršiti ažuriranje težina klauza.

3.9.3 FlipGA

FlipGA (*Using the Flip Heuristic*) su predstavili Marchiori i Rossi [3]. Ovo je zapravo klasičan evolutivni algoritam 3.1 koji je unapređen putem lokalne pretrage 3.8. Koristimo klasičnu funkciju prilagođenosti 3.3 i ruletsku selekciju 3.4.1.

FlipGA ima populaciju veličine 10, koristi generacijsku politiku zamene 3.7 sa strategijom elitizma, koja ostavlja najbolje 2 jedinke iz trenutne generacije u narednu. Ovo je jedini algoritam u kojem uvek vršimo uniformno ukrštanje 3.5 i klasičan operator mutacije 3.6.1 u kojem parametar mutiranja uzima vrednost 0.9.

3.9.4 ASAP

ASAP (*Using the Flip Heuristic and Adaptation*) je varijanta ASAP [3]. Nastala je od ASAP ali sa drugačijom politikom zamene. Naime, ovde koristimo $(1 + 1)$ strategiju zamene 3.7 (1 roditelj proizvodi 1 dete). Iz ovoga vidimo da je populacija veličine 1, zato ukršanje nije moguće.

Kreiranje nove generacije se vrši na sledeći način: uzmemo jedinog mogućeg roditelja, slučajno prilagodljivom mutacijom 3.6.2 dobijamo jedno dete. Primenimo lokalnu pretragu sa korišćenjem tabu listi 3.8. Postoji $(1 + 1)$ u pitanju, nova generacija je ili roditelj ili dobijeno dete, u zavisnosti od toga ko je prilagođeniji. Koristimo najobičniju funkciju prilagođenosti 3.3.

3.10 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.

Tabela 2: Parametri

Parametar	Vrednost
Broj iteracija	1000
max flip	30000

Tabela 3: AIM nezadovoljivi testovi

Instanca	Broj literala	Broj klauza	SAWEA	RFEA	FlipGA	ASAP
aim-50-1_6-no	50	80	79	79	79	79
aim-50-2_0-no	50	100	98.6	99	99	99
aim-100-1_6-no	100	160	158.8	159	159	159
aim-100-2_0-no	100	200	197.8	199	199	199
aim-200-2_0-no	200	400	396	397.8	399	399

4 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.

Tabela 4: AIM zadovoljivi testovi

Instanca	Broj literala	Broj klauza	SAWEA	RFEA	FlipGA	ASAP
aim-50-1_6-yes	50	80	79	79	79	79
aim-50-2_0-yes	50	100	98.4	99	99	99
aim-50-3_4-yes	50	170	167	167.2	170	170
aim-50-6_0-yes	50	300	300	290	300	300
aim-100-1_6-yes	100	160	158.2	158	159	159
aim-100-2_0-yes	100	200	195.8	197.8	199	199
aim-100-3_4-yes	100	340	324.6	332.4	340	340
aim-100-6_0-yes	100	600	586.8	576	600	600
aim-200-2_0-yes	200	400	392.2	396	399	399
aim-200-6_0-yes	200	1200	1133	1173	1200	1200

Optimizacija rojem čestica je algoritam zasnovan na ponašanju pojedinačnih jedinki unutar određene grupe (na primer, jata ptica ili roja insekata). Ukoliko se, vođeno instinktom, 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 instinktom 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:

$$\vec{v}_i^{t+1} = w \cdot \vec{v}_i^t + c_1 \cdot \vec{r}_1 \times (\vec{P}_i^t - \vec{x}_i^t) + c_2 \cdot \vec{r}_2 \times (\vec{P}_g^t - \vec{x}_i^t) \quad (1)$$

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1} \quad (2)$$

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 [4], dat je sigmoidnom transformacijom. Sada v_i^t predstavlja verovatnoću da bit x_i^t uzme vrednost 1.

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

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

4.1 Pseudokod PSO

U ovom poglavlju dat je osnovni oblik algoritma na kojem se zasnivaju ostale varijante i izmene koje će biti detaljnije izložene. Jednu populaciju, odnosno roj, čini unapred određen broj čestica, lista potencijalnih rešenja kao i dodeljene brzine za svaku od njih. Kroz iteracije računa se fitness, ažuriraju se brzine i pozicije čestica dok se ne zadovolji kriterijum zaustavljanja. Jedna varijanta (WPSOSAT) uvodi i lokalnu pretragu koja se izvodi umesto ažuriranja pozicija, odnosno jednačine 3. 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. U test primerima za koje unapred znamo da je formula zadovoljiva, ili koliko je klauza zadovoljivo, možemo koristiti kriterijum da se dostigao ukupan broj zadovoljivih klauza.

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

inicijalizacija populacije: pozicije i brzine;
 $t = 0$; // **tekuća iteracija**
while *nije zadovoljen uslov zaustavljanja* **do**
 $t = t + 1$;
 for $i \leftarrow 0$ **to** *broj čestica u roju* **do**
 Izračunaj $\text{Fitness}(\vec{P}_i^t)$;
 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 2: Osnovni PSO algoritam

4.2 Inicijalizacija rešenja

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.

4.3 Fitnes funkcija

Fitnes funkcija je veoma važna za performanse algoritma. Prva fitnes funkcija koja se sama nameće jeste broj zadovoljenih klauza, kakva je data u samoj formulaciji problema, ali se takva funkcija nije pokazala kao dovoljno dobra. Bolji mehanizam je stepenasto ažuriranje težina (SAW - stepwise adaptation weights) uvedena od strane Eiben-a [2]. Ona je data sledećim formulama:

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

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

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.

4.4 Lokalna pretraga i flip heuristika

Da bi se unapredio standardni PSO algoritam uvedena je, umesto ažuriranja pozicija čestice jednačinom 3, stohastička lokalna pretraga heuristikom okretanja bitova (eng. *flip heuristic*). Ova heuristika zasniva se na izmeni pojedinačnih bitova u tekućem rešenju. Za svaki bit u čestici pokušava se njegovo okretanje. Ako sa tom izmenom ima dobiti (u smislu ne pogoršavanja funkcije cilja) čuva se nova vrednosti okrenutog bita, dok se u suprotnom vraća na njegovu staru vrednost. Čitav proces se ponavlja dokle god ima napretka, ako je bar jedna izmena poboljšala tekuće rešenje.

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;
            improvement += gain;
        end
        else
            odbaci flip, vrati na staru vrednost p[i];
        end
    end
end
end

```

Algorithm 3: Funkcija lokalne pretrage

4.5 Varijante PSO algoritma

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

4.5.1 PSO-LS

4.5.2 PSOSAT

4.5.3 WPSOSAT

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 raznanaže 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.

4.6 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 6 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 7 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.

Tabela 5: Parametri

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 6: AIM nezadovoljivi testovi

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-no	200	400	398.6	399	399

Tabela 7: AIM zadovoljivi 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

5 Zaključak

Literatura

- [1] Benno Stein Hans Kleine Buning. A study of evolutionary algorithms for the satisfiability problem, 2004.
- [2] Gottlieb J and Voss N. Adaptive fitness functions for the satisfiability problem. 2000. In: Parallel Problem Solving from Nature - PPSN VI 6th International Conference, Paris, France. Springer Verlag. LNCS 1917.
- [3] Claudio Rossi Jens Gottlieb, Elena Marchiori. Evolutionary algorithms for the satisfiability problem.
- [4] Eberhart RC. Kennedy J. A discrete binary version of the particle swarm algorithm, 2007.
- [5] Mladen Nikolić Predrag Janičić. *Veštačka inteligencija*. Beograd, 2019.