MAXiMUM SATiSFYiNG LiNEAR SUBSYSTEM

Studentski projekat

Milica Nikolić

Sadržaj

1 Uvod....3
2 Vizuelizacija problema....4
3 Težina razmatranog problema....5
3.1 Težina MSLS problema....5
3.2 Srodni problemi....7
4 Aproksimacija particionisanjem....8
4.1 Aprkosimacija problema....8
4.2 Naslednosti....8
4.3 Particionisanje u manje podskupove....9
5 Opis rešenja problema....10
5.1 Linearno programiranje – egzaktni pristup....10
5.2 Genetski algoritam – aproksimativni pristup....11
5.2.1 Motivacija....11
5.2.2 Opšte karakteristike – reprezentacija i operatori....11
5.2.3 Inicijalizacija i funckija prilagođenosti....12
5.2.4 Kriterijum zaustavljanja....12
5.2.5 Grafik broja jendačina u rešenju u toku vremena....12
5.2.6 Evaluacija modela....13
6 Zaključak....14
4 Literatura....15

1 Uvod

Razmatramo problem ***Maximum Satisfying Linear Subsystem (MSLS)*** definisan na sledeći način:

Zadat je sistem Ax=b linearnih jednačina, gde je A matrica celih brojeva dimenzija mxn, a B vektor celih brojeva dužine m. Resenje problema je vektor racionalnih brojeva x dužine n, kojim je zadovoljen najveci broj datih linearnih jednčina.

Kada zadati sistem Ax=b ima rešenje, problem je trivijalan i može se rešiti u polinomskom vremenu Gausovom metodom eliminacije (ili Kramerovim pravilom). U slučaju kada zadati sistem nema rešenje, pronalazak vektora x koji zadovoljava maksimalno mnogo datih jednačina je NP-težak problem.

Pokazaćemo da se MSLS problem može aproksimirati u O(m/logN), ali da se ne može aproksimirati u m^{ε} za svako ε, ε>0, gde je m broje jednačina u sistemu. Teorijski ćemo se osvrnuti na pojmove naslednosti i praticionisanja.

Problem će biti rešen na egzaktna i na približan/aproksimativan način zbog kombinatorne eskplozije za probleme velikih dimenzija.

2 Vizuelizacija problema

Najpogodniji sistem linearnih jednačina, u cilju vizuelizacije problema koji razmatramo, je onaj sa dve dimenzije. Odnosno sistem oblika:

$a_1 X + b_1 Y = c_1$

$a_2 X + b_2 Y = c_2$

…

$a_m X + b_m Y = c_m$

gde su a_i, b_i, c_i elementi skupa celih brojeva, $0<i<m$, a X i Y nepoznati parametri, čije je vrednosti potrebno naći, a tako da što veći broje linearnih jednačina bude zadovoljen.

Razmotrimo sada grafikon na kom su nacrtani grafikoni 12 linearnih jednačina koje čine jedan sistem (Figue 1).

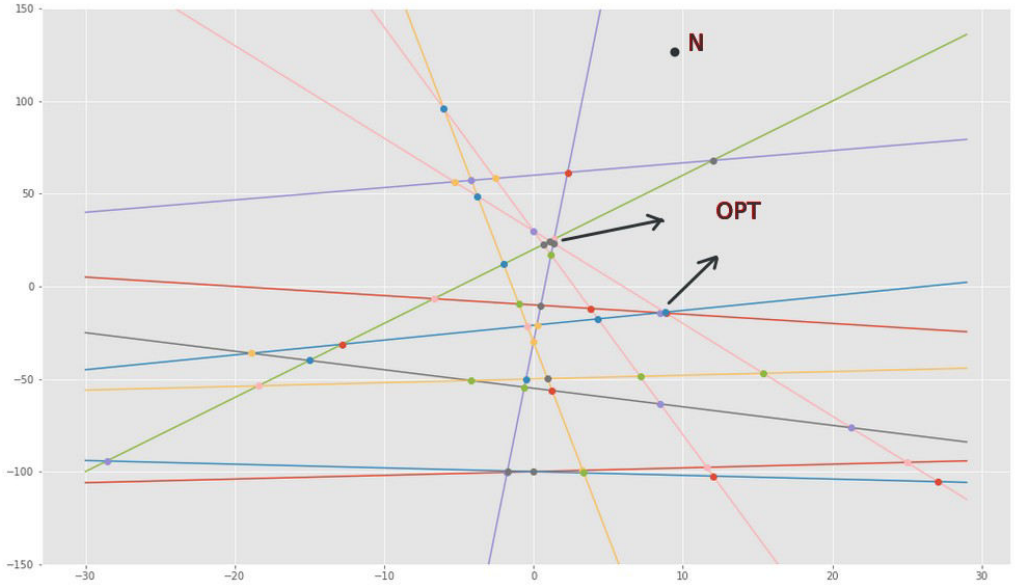


Figure 1: Grafikon 12 pravih

Ono što uočavamo jesu dva ***optimalna rešenja problema*** – tačke OPT koje zadovoljavaju tri jednačina iz sistema, jedno ***nedopustivo rešenje*** – tačaka N, veliki broj ***dopustivih rešenja*** – sve ostale tačke na priloženom grafikonu. U konkretnom slučaju za dve dimenzije, skup dopustivih rešenja su sve tačke koje pripadaju barem jednoj pravoj zadatom jednačinom sistema. Optimalno rešenje je ona tačka koja se nalazi na najvećem broju ovakvih pravih. Svako drugo rešenje je nedopustivo. Slično definišemo i pojmove u više dimenzija (kada imamo jednačine sa više od dve nepoznate), osim što će prava posatje funkcija više promenljivih, pa ih to čini nepodobnim za vizuelizaciju.

3 Težina razmatranog problema

Pre nego što počnemo sa rešavanjem bilo kakvog problema, potrebno je proceniti koliko je razmatrani problem težak. Na osnovu toga zaključujemo da li je problem moguće rešiti nekom egzaktnom metodom ili je neophodna upotreba približnog (aproksimativnog).

3.1 Težina MSLS problema

Problem koji razmatramo, MSLS, je NP-težak problem i pripada grupi problema maksimalnih nezavisnih skupova (***maximal independent set (MIS)*** *ili* ***maximal stable set***). U nastavku razmatramo u koje još klase složenosti spada MSLS i neke njegove varijante.

Definicija 2.0.1. [4] NP problem optimizacije F je ***polinomski ograničen*** ako postoji polinom p takav da: $\forall x \in P_F \ \ \forall y \in S_F(x), m_F(x, y) \leq p(|x|)$
Gde je P_F prostor ulaznih instanci problema F, $S_F(x)$ prostor dopustivih rešenja, a $m_F(x, y)$ funkcija cilja.

Klasa polinomski ograničenih NP problema optimizacije zove se ***NPO BP***, i u nju spada i razmatrani MSLS problem.

Postoje NPO problemi koji se mogu aproksimiratu u konstanti, dok neki problemi ne mogu biti aproksimirani ni u n^{ε} za svako $\varepsilon > 0$, gde je n velicina ulazne instance, ukoliko ne važi da je N=NP. Negde između, nalaze se problemi koji pripadaju ***klasi APX*** i mogu se aproksimirati u nekoj konstanti.

Narednu teoremu navodimo bez dokaza:

Teorema 2.0.1. [4] Problem MSLS je APX-težak čak i kada uvedemo ograničenje da je homogen i sa dikretnim koeficijentima iz skupa {-4, -3, -2, -1, 0, 1, 2 , 3, 4} i bez parova identičnih jednakosti.

Od velike važnosti je propozicija koja se oslanja na prethodnu teoremu.

Propozicija 2.0.1. [4] Problem MSLS ne pripada klasi APX osim ukoliko ne važi da je P=NP.

U nastavku ćemo dokazati prethodnu propoziciju:

Pretpostavimo da možemo aproksimirati MSLS u konstanti c, c >1 i razmotrimo proizvoljnu instancu sa p homogenih jednačina $e_1=0$, $e_1=0$, … , $e_p=0$. Neka je s broj jednačina u maksimalnom saglasnom podskupu ovog sistema. Posmatrajmo sada nvi problem sa jednačinama $e_{i,\ j,\ k}=0$, gde je $e_{i,j,k}=e_i+ke_j$, $1\leq i\leq p$, $1\leq j\leq p$, $1\leq k\leq T$ za celobrojno T. Kako su $e_i=0$ i $e_j=0$, sledi i da je $e_{i,j,k} = 0$ za svaku vrednost k. Broj zadovoljenih jednačina novog problema koji razmatramo je Ts^2. Međutim, neke dodatne jednačine mogu biti zadovoljene u slučajevima kada je $e_i=-ke_j$ i $e_i\neq 0$. Takvih jednačina ne može biti više od p^2. Kako optimalno rešenje sadrži barem Ts^2 zadovoljenih jednačina,

aproksimacioni algoritam obezbeđuje rešenje koje zadovoljava barem Ts^2/c jednačina. Ispitivanjem zadovoljenih jednačina odbacujemo one z akoje važi da je $e_i=-ke_j$ i $e_i\neq0$. To nam ostavlja barem $Ts^2/c - p^2$ jednačina. Kako imamo najviše T jendačina za svaki par (i, j), dobijamo najviše:

$$\sqrt{(Ts^2/c - p^2)}/\sqrt{T} = \sqrt{(s^2/c - p^2/T)}$$

zadovoljenih jednačina početnog problema. Pokretanjem aproksimacionog algoritma direktno na originalnom problemu mi garantujemo pronalaženje s/c zadovoljenih jednačina. Biranjem

$$T \geq \lceil (p^2c^2)/(s^2(c-1)) \rceil + 1$$

više jednačina je zadovoljeno primenivanjem aproksimacionog algoritma na $e_{i,j,k}$ problem nego primenom na originalni problem. Kako iz Teoreme 2.0.1. sledi da je MSLS AXP-težak problem i stoga postoji konstanta β, 0< β <1, takva d a aproksimacija ne može biti u konstanti manjoj od 1/(1-beta). S toga MSLS ograničen samo na homogene jednačine nije u APX, pa samim tim ***MSLS ne pripada APX klasi problema.***

Korišćenjem torki od log p jednačina umesto parova u prethodnom postupku možemo doći i do jačeg zaključka:

Teorema 2.0.2. [4] Ukoliko ne važi da je P=NP, postoji pozitivna konstanta ε takva da se homogeni MSLS ne može aproksimirati u p^{ε}, gde je p broj jednačina.

Dokažimo navedenu teoremu:

Pretpostavimo da možemo aproksimirati MSLS u konstanti c, c >1 i razmotrimo proizvoljnu instancu sa p homogenih jednačina $e_1=0$, $e_2=0$, … , $e_p=0$. Neka je s broj jednačina u maksimalnom saglasnom podskupu ovog sistema. Pretpostavimo da fiksan procenat β jednačina može biti zadovoljen. β možemo uvećati dodavanjem trivijalnih zadovoljivih jednačina u sistem. Posmatrajmo sada m-torke jednačina ovog sistema, gde je m približno jednako log p. Razmotrimo sledeće:
$e_{i,k} = \Sigma e_{ij}k^j$ za j=1, …, j=m i $1\leq i\leq N$, $1\leq k\leq T$

za celobrojne N, T koje treba odrediti. Za svako i za koje nisu svi $e_{ij}=0$ polinomska jendačina $\Sigma e_{ij}x^j$ ima najviše m rešenja od kojih je jedno x=0. Stoga najviše m-1 od T jednačina $e_{i,k}+0$, $1\leq k\leq T$ može biti zadovoljeno ukoliko je $e_{ij}=0$ za svako j u [1,…, m]. m-torku za koju je svako $e_{ij}=0$ zovemo *dobra torka.*

Ovde je problem to što ne možemo oformiti nove jednačine od svake torke starih jednačina u polinomskom vremenu i tako da broj dobrih torki u skupu i podskupu bude srazmeran (Za detalje pogledati reference iz [4]).

Zaključujemo dakle da postoje ε, ε>0, takve da se problem koji razmatramo ne može aproksimirati u p^{ε}.

3.2 Srodni problemi

U nastavku navšćemo kompleksnosti aproksimacija problema srodnih onom koga razmatramo.

Nadklasa problema kog razmatramo je MAX FLS problem definisan na sledeći način:

MAX FLS problem je problem pronalaska maksimalnog saglasnog podskupa zadatog skupa linearnih relacija (≤, =, ≠, ≥).

MAX FLS^{Rel} problem, gde Rel∈{≤, =, ≠, ≥}, je problem definisan na sledeći način: dat je linarni sistem Ax Rel b, sa matricom A - pxn, pronaći x∈R^n koje zadovoljava što je više moguće datih relacija.

U ovakvoj notaciji MSLS problem bio bi ***MAX FLS$^{=}$*** problem sa ograničenjem da su koeficijenti celi brojevi.

	Real variables	Binary variables
MAX FLS$^{=}$	Not within p^{ε} for some $\varepsilon > 0$	MAX IND SET-hard
MAX FLS$^{\geq}$	APX-complete (within 2)	MAX IND SET-hard
MAX FLS$^{\neq}$	Trivial	APX-complete (within 2)
C MAX FLS$^{=;=}$	Not within p^{ε} for some $\varepsilon > 0$	NPO PB-complete
C MAX FLS$^{=;\geq}$	APX-complete (within 2)	NPO PB-complete
C MAX FLS$^{\geq;=}$	MAX IND SET-hard	NPO PB-complete
C MAX FLS$^{\geq;\geq}$	MAX IND SET-hard	NPO PB-complete
C MAX FLS$^{\neq;=}$	MAX IND SET-hard	NPO PB-complete
C MAX FLS$^{\neq;\geq}$	APX-complete (within 2)	NPO PB-complete
C MAX FLS$^{\mathcal{R};\neq}$	Trivial	NPO PB-complete

[4] Main approximability results for MAX FLS variants.

4 Aproksimacija particionisanjem

Razmotrićemo kako se problemi pronalaska maksimalnog podgrafa/podskupa mogu aproksimirati particionisanjem zadatog grafa/skupa u jednostavnije podprobleme. Ramotrićemo moguće performanse problema maksimizacije u kojima je bilo koji podskup rešenja, takođe rešenje problema.

Važno je napomenuti da, osobine i teoreme koje su navedene u nastavku jesu u *jeziku grafova*, ali generalizuju se i na ostale diskretne strukture podataka.

4.1 Aprkosimacija problema

Aproksimirati neki optimizacioni problem znači pronaći bilo koje dopustivo rešenje. Koliko je dobra aproksimacija zavisi od relacije između optimalnog rešenja i rešenja koje se može dobiti aproksimacijom. idealno, aproksimacija je optimalna do malog faktora (na primer 5%). Za neke aproksimacione algoritme moguće je dokazati odgovarajuće osobine optimalnog rešenja. Pa je ***p*-aproksimacioni** algoritam A algoritam za koji je dokazano da vrednost f(x) aproksimativnog rešenja A(x) neće biti veća (ili manja) od pOPT, gde je OPT vrednost optimalno rešenja.

Dakle važi:

OPT≤ f(x) ≤ pOPT, za p >1

pOPT≥ f(x) ≥ OPT, za p ≤1

Faktor *p* naziva se *relativni faktor optimizacije* i što je *p* bliže jedinici, aproksimacija je bolja.

4.2 Naslednosti

Osobina nekog grafa je *naslednost* ukoliko važi da ako graf ima tu osobinu, ima je i svaki njegov indukovani podgraf. Kako se ova osobina može generalizovati i na ostale strukture podataka, važi sledeće: osobina skupa je ***naslednost*** ako važi da ako skup ima tu osobinu, ima je i svaki njegov podskup.
Primetimo da je osobina ***saglasnosti*** skupa linearnih jednačina naslednost.

Uvedimo sledeću lemu:

Lema 2.1.1. [1] Neka je Π problem pronalaženja maksimalnog indukovanog podgrafa koji zadovoljava naslednu osobinu π. Ako možemo da particionišemo graf G u podgrafove G1, G2, …, Gt i rešimo Π optimalno za svaki podgraf Gi, onda možemo aproksimirati Π na G unutar t.

U nastavku navodimo opštiju teoremu:

Teorema 2.1.1. [1] Neka je Π problem pronalaženja maksimalnog indukovanog podgrafa koji zadovoljava naslednost π. Pretpostavimo da možemo:

- izvući indukovne podgrafe grafa G: G1, G2, … , Gt takve da je svaki čvor iz G sadržan u barem k različitih podgrafa Gi
- pronaći zadovoljivo rešenje problema Π - HEU(G), takvo da HEU(Gi)*pi ≥ OPT(Gi)

Tada,

$$HEU(G) + \max_i HEU(Gi) \geq OPT(G)*k/(\sum pi)$$

4.3 Particionisanje u manje podskupove

Za osobinu grafa kažemo da je ***eksponencijalo proverljiva* (*EXP-checkable*)** ako za zadati graf od n čvorova, osobina može biti proverena u složenosti najviše 2^{n^c} gde je c konstanta. U klasu ovih osobina spadaju mnoge osobine grafova kao što su planarnost, k-obojivost, savršen graf, itd.

Ako pretpostavimo da možemo proizvoljno particionisati graf u n/logn skupova, svaki sa po logn čvorova. Tada jednostavnim algoritmom grube sile možemo pronaći optimalno rešenje bilo kog problema maksimalnog podgrafa koji ima svojstvo naslednosti, u polinomskom vremenu.

Uvedimo još dve leme:

Lema 2.2.1. [1] Svaki problem pronalaženja težinskog indukovanog podgrafa koji zadovoljava eksponencijalo proverljivu (*EXP-checkable*) naslednost može se aproksimirati u n/logn (n/logn-aproksimacija).

Pokazuje se da je n/logn za sada najbolja aproksimacija ovakvih problema. Problem podgrafa za bilo koju netrivijalnu naslednost nije moguće aproksimirati u c, gde je c konstanta, osim ukoliko ne važi da je N=NP. Ovo se može generalizovati i za neke probleme podskupa, jedan od njih je MSLS problem.

Lema 2.2.2. [1] Neka je Π problem pronalaženja maksimalnog podskupa koji zadovoljava naslednost π za čije se instance mere najviše logn, π može ispitati u polinomskom vremenu. Tada, problem se može aproksimirati u vremenu O(n/logn).

Problem zadovoljivosti sistema linearnih jednačina je moguće rešiti u polinomskom vremenu linearnim programiranjem.

Dakle, Maximum Satisfying Linear Subsystem (MSLS) problem može se uproksimirati u O(n/logn), tačnije O(n/logN), gde je N veličina ulaza.

5 Opis rešenja problema

Problem je rešen na dva načina: linearnim programiranjem i genetskim algoritmom. Oba rešenja napisana su u programskom jeziku Python. Algoritmi su testirani nad nasumično kreiranim sistemima raznih dimenzija sa celobrojim koeficijentima iz skupa [-5000, 5000]. Kreiranje pomenutih sistema tako]e je relizovano u Python-u i svaki test primer je zapisan u datoteci sa ekstenzijom .txt u priloženom folderu tests.

5.1 Linearno programiranje – egzaktni pristup

Jedan on načina da se ovaj problem reši jeste primenom linearnog programiranja. Ovaj algoritam je egzaktan, odnosno pronalazi optimalno rešenje MSLS problema. Implementacija je u programskom jeziku Pyhon i u fokusu je biblioteka linprog iz scipy-a.

Ograničenja problema linearnog programiranja zadajemo kao zadovoljivost jednačina koje šaljemo rešavaču, a one su predstavljene u matričnom obliku.

U ovom slučaju, naivni pristup bi bio ispitivanje zadovoljivosti svih kombinacija jednačina zadatog sistema. Primena bi mogla biti odozgo – počeli bismo od kombinacije dužine n, gde je n broj jednačina ili odozdo – počeli bismo od svih kombinacije dužine dva, zatim tri, i tako redom.

Međutim, mi smo pokazali da je zadovoljivost sistema linearnih jednačina naslednost. Odnosno, ako je neki sistem zadovoljiv, zadovoljiv je svaki njegov podsistem. Ovu važnu osobinu možemo iskoristiti u smislu da kada ustanovimo da neki sistem jednačina nije zadovoljiv, sigurno neće biti zadovoljiv bilo koji njegov nasistem, pa ga nije neophodno ispitivati.

Ideja je da svaki nezadovoljiv podsistem označimo kao nezadovoljiv i ubuduće ne ispitujemo zadovoljivost bilo kog sistema koji ga sadrži. To lako možemo postići izbegavanjem kreiranja već pomenutih kombinacija koje sadrže označene podsisteme.

Ovakav pristup, iako bolji od naivnog, u opštem slučaju je i dalje eksponencijalne složenosti. Kako je takva složenost neprihvatljiva za probleme većih dimenzija, zaključujemo da je potrebno napraviiti i algoritam zasnovan na približnom (aproksimativnom) pristupu.

Ovakav pristup ostvaren je genetskim algoritmom, koji će za probleme manjih dimenzija biti upoređen sa prethodno opisanim egzaktnim pristupom.

5.2 Genetski algoritam – aproksimativni pristup

Genetski algoritmi (GA) su nastali u Americi 1970-ih odina. GA su podgrana računarske inteligencije (RI) i pripadaju klasi Evolutivnih algoritama (EA). Predstavljaju pretraživačku heuristiku koja oponaša proces prirodne selekcije. U GA, kroz ukrštanje i mutaciju, važnu ulogu imaju slučajni događaji. Na ovaj način, kroz proces prirodne selekcije, ukrštanje i mutaciju, GA, kao i svi EA, imitira proces prirodne evolucije.

5.2.1 Motivacija

Pitanje koje se prvo nameće je: Zašto koristiti GA u rešavanju MSLS problema?

Glavne primene genetskih algoritama su u diskretnim domenima i pokazali su se veoma dobrim za rešavanja kombinatornih problema. MSLS se svodi na pronalaženje kombinacija jednačina iz datog sistema koje čine zadovoljiv sistem. Pri rešavanju ovog problema velikih dimenzija dolazi do kombinatorne eksplozije, pa se odlučujemo da problem aproksimiramo genetskim algoritmima koji su se za to pokazali dobrim.

Takođe, jedna od mana svih EA je teško kodiranje (reprezantacija) rešenja problema – hromozoma. Međutim, za razmatrani problem, već intuitivno se nameću neki od načina kodiranja – binarni (1 za jednačino koja se nalazi u podsistemu, 0 za jednačinu koja se ne nalazi) i celobrojni (redni brojevi jednačina koje se nalaze u podsistemu).

5.2.2 Opšte karakteristike – reprezentacija i operatori

U nastavku se mogu videti osnovne karakteristike primenjenog algoritma:

Karakteristike GA	Implementacija
Reprezentacija	Binarni n – dimenzioni vektor gde je n broj jednačina u sistemu. 1 označava da je jednačina sadržana u sistemu, 0 označava da nije
Ukrštanje	Kombinacija jednopozicionog i ravnomernog ukrštanja
Mutacija	Izvrtanje jednog bita
Selekcija roditelja	Turnirska selekcija
Selekcija preživelih	Selekcija sa stabilnim stanjem, s korišćenjem elitizma

Kao reprezentacija hromozoma razmatrana je i celobrojna varijanta, ali kako to komplikuje operatore ukrštanja i mutacije, opredeljujemo se za binarnu. Testiranjem raznomernog, jednopozicionog i višepozicionog ukrštanja, ravnomerno se ispostavilo kao neefikasno za ovaj problem.

Selekcija roditelja rađena je turnirski, a selekcija preživalih realizovana je kao potpuna zamena roditelja decom. Uz to, korišćen je i elitizam, pa je K najboljih jedinki uvek zadržavano u narednoj populaciji. K smo postavljali tako da čini oko 10% populacije.

Operator mutacije je naravno prisutan, ali je verovatnoća da se mutacija dogodi vremenom, kroz iteracije opada.

5.2.3 Inicijalizacija i funckija prilagođenosti

Prvi korak svakog GA je Inicijalizacija početne populacije. Početna populacija u GA , primenjenom na MSLS problem, se sastoji od podsistema dužine n, gde je n broj nepoznatih. Ono što odmah primećujemo je da može da se desi da u početnoj populaciji ***postoje jedinke koje predstavljaju nedopustiva rešenja problema***. To naravno, nismo uradili slučajno. Naime, MSLS je problem u kom nedopustivo rešenje može biti veoma blizu nekog veoma dobrog dopustivog rešenja. Razmotrimo situaciju u kojoj jedinka predstavlja nesaglasan sistem od K jednačina. Može se desiti da ovaj sistem postaje saglasan odstranjivanjem k jednačina iz sistema, gde je k znatno manje od K. Mi optimistično pretpostavljamo da će se ovaj nesaglasan sistem, ukrštanjem i mutacijom, postati saglasan, tj. da će u budućnosti imati potomke koji predstavljaju kvalitetna dopustiva rešenja.

Kako smo dopustili jedinkama koje predstavljaju nesaglasna rešenja da žive u našoj populaciji, bilo je neophodno i napraviti ***odgovorajuću funckiju prilagođenosti*** (fitness function – FF). Naime, funkcija će proveravati da li je podsistem koji jedinka kodira saglsan ili ne. Ukoliko jeste, vraća rešenje tog sistema, a ukoliko nije, vraća razmeru broja jednačina i broja promenljivih (alternativa bi bila da vraća zbir kvadriranih grešaka za svaku od jednačina u odnosu na najbolje pronađeno rešenje nekom lokalnom pretragom). Ovo je implementirno, ponovo, linearnim programiranjem, simplex metodom.

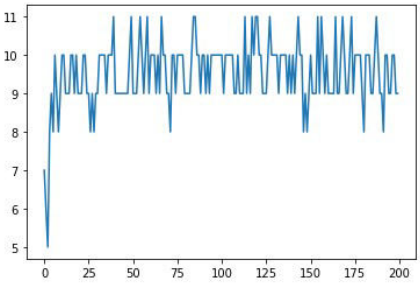
5.2.4 Kriterijum zaustavljanja

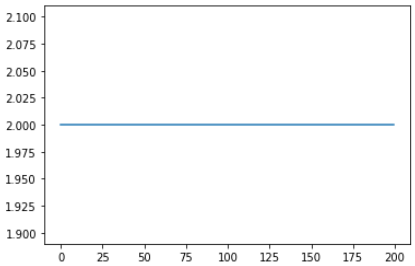
Dakle, inicijalizovali smo početnu populaciju, definisali funkciju prilagođenosti, operatore, način selekcije roditelja i način selekcije preživalih. Svi prethodno definisani pojmovi zajedno u petlji čine jedan GA. Postavlja se pitanje kada zaustaviti algoritam?

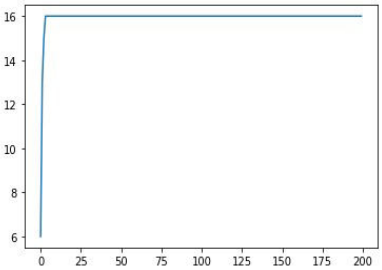
U našoj implementaciji kriterijum zaustavljanja je zasnovan na maksimalnom broju iteracija – maxIter.

5.2.5 Grafik broja jendačina u rešenju u toku vremena

U nastavku su prikazana tri grafika na kojima se vidi kako se broj jednačina u najboljem rešenju menja u toku iteracija.

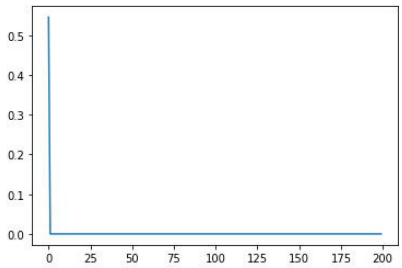


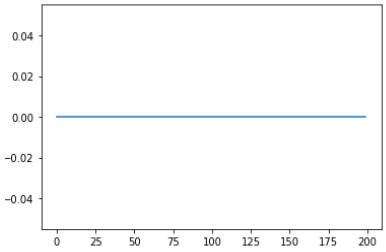


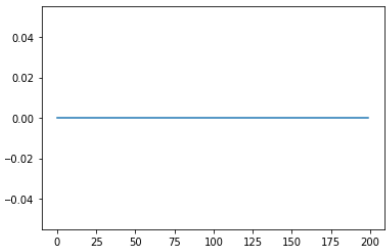


Razlog nepravilnosti prvog grafika je to što smo dopustili algoritmu da istražuje i nedopustiva rešenja, koja naravno mogu sadržati više jednačina od nekih dopustivih. Čim se pojavi rešenje bolje prilagođenosti, ono postaje optimalno, iako sadrži manje jednačina od prethodnog, što dovodi do grafika nepravilnih oblika.

U nastavku su i tri grafikona koja predstavljaju vrednosti stepena nedopustivosti koja je obrnuto srazmerna prilagođenosti rešenja u toku vremena.







5.2.6 Evaluacija modela

U nastavku poredimo rešenja dobijena egzaktnim pristupom i primenjenim genetskim algoritmom za probleme malih dimenzija. Za probleme velikih dimenzija samo prikazujemo rešenje dobijeno primenom genetskog algoritma.

U genetskom algoritmu je parametar maxIter postavljen na 200.

Dimenzija ulaza	Rešenje dobijeno egzaktnom metodom	Rešenje dobijeno genetskim algroritmom
4x4	2	2
7x4	5	5
17x4	16	16
25x5	5	5
5x20	5	5
35x4	4	4
38x4	7	5
39x4	6	6
8x30	8	8
50x11	Nije mereno	10
99x10	Nije mereno	2
100x50	Nije mereno	37
200x5	Nije mereno	3
260x4	Nije mereno	173

6 Zaključak

Za rešavanje NP teških problema, kao približne metode, neretko se koristi genetski algoritmi, ili neki drugi algoritmi koji pripadaju grupi evolutivnih algoritama. Pored dobro izabrane reprezentacije, operatora ukrštanja i mutacije, kriterijuma zaustavljanja, od velike je važnosti izabrati pogodnu funkciju prilagođenosti. Kod kombinatornih problema, odnosno probema sa diskretnim domenima, a koji uključuju ograničenja može se desiti da je skup dopustivih rešenja veoma mali. Takav je i MSLS problem koji razmatramo. Istraživati samo dopustiva rešenja u ovakvim situacijama može biti neefikasno, pa nadalje pretpostavljamo da se istražuju i nedopustiva rešenja. Ako funckiju prilagođenosti definišemo tako da nedopustiva rešenja smatra potpuno neprilagođenim i među njima ne pravi nikakvu razliku, javljaju se razni problemi. Jedan od ključnih je to što u fazi selekecije, operator neće praviti razliku između rešenja koje je nedopustivo, ali se sitnim izmenama može korigovati u dopustivo i nedopustivih rešenja za čije su korigovanje potrebne krupnije izmene. Kao rešenje toga, možemo uvesti pojam stepena nedopustivosti i funckiju prilagođenosti napraviti tako da nedopustiva rešenja sa manjim stepenom nedopustivosti imaju bolju prilagođenost. U GA koji je implementiran u ovom projektu je u te svrhe, u funkciji prilagođenosti, korišćena razmera pomenuta u prethodnom poglavlju.

Pri konstrukciji približnih metoda uvek treba imati na umu prioritet optimizacije – da li je to što brži pronalazak rešenja, ili na štetu vremenske efikasnosti želimo da pronađemo kvalitetnije rešenje. Ovim ciljevima, u EA, uvek su prilagođeni operatori i funckija prilagođenosti.

Kvalitet svakog približnog algoritma preporučljivo je oceniti poređenjem dobijenih rešenja sa rešenjima dobijenim nekom egzaktnom metodom ili rešenjima iz literature.

4 Literatura

(1) Magnús M. Halldórsson. Approximations via Partitioning (1995)

(2) Zeev Nutov, Daniel Reichman. Approximating maximum satisfiable subsystems of linear equations of bounded width (The Open University of israel)

(3) Arora, S., Babai, L., Stern, J., and Sweedyk, Z. The hardness of approximate optima in lattices, codes, and systems of linear equation' (1997)

(4) Amaldi, E., and Kann, V. The complexity and approximability of finding maximum feasible subsystems of linear relations (1995)

(5) Materijali korišćeni na kursu Računarska Inteligencija

(6) Predrag Jančič, Mladen Nikolić Veštačka inteligencija