SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINARSKI RAD
IZ KOLEGIJA
ALGORITMI U SUSTAVIMA UPRAVLJANJA

USPOREDBA HEURISTIČKIH ALGORITAMA NA PROBLEMIMA OPTIMIRANJA, NAPRTNJAČE I PROBLEMU n-KRALJICA

Ivica Martinjak

Zagreb, ožujak 2006.

Sadržaj:

1. Uvod

Determinističke tehnike ne daju rješenje NP teških[1] problema u prihvatljivom vremenskom okviru, obzirom na njihovu gornju granicu složenosti. Rješenje ovog problema predstavljaju heuristički algoritmi, čijom se primjenom NP klasa problema može uspješno rješavati u prihvatljivom vremenskom okviru.
Osnovna karakteristika heurističkih algoritama je nesigurnost da li je dobiveno rješenje optimalno. Međutim, pokazalo se da iako nema garancije da je generirano rješenje najbolje moguće, heuristički algoritmi s visokom vjerojatnošću daju optimalno rješenje ili rješenje blisko optimalnom (obično 2-5% odstupanja).

Teškoća pri korištenju heurističkih algoritama je kreiranje uspješne heuristike te pronalaženje optimalnih parametara za dani problem. Heuristika je vitalni dio svakog heurističkog algoritma, o kojem najviše ovisi uspješnost algoritma a čije formiranje nije standardizirano već zahtijeva kreativnost i iskustvo[2]. Može se reći da jedino genetski algoritam ima univerzalnu heuristiku – oponašanje prirodnog evolucijskog procesa, no i pri tome je puno stupnjeva slobode.

U ovom se radu heuristički algoritmi uspoređuju na trima primjerima. Kao prvo su osmišljene i implementirane heuristike za rješavanje odabranih problema[3] a zatim su napravljeni usporedni testovi. Penjanje uzbrdo, simulirano kaljenje, tabu pretraživanje i genetski algoritam uspoređeni su na primjeru optimiranja realne funkcije jedne varijable. Algoritmi simulirano kaljenje, tabu pretraživanje i genetski algoritam uspoređeni su na problemu naprtnjače i na problemu n-kraljica.

U prvom dijelu rada opisuju se uspoređivani heuristički algoritmi s naglaskom na mogućnosti i načine njihove primjene. U drugom poglavlju opisuju se razvijene heuristike za optimiranje funkcija, rezultati testiranja kao i izvedeni zaključci. Na isti način prezentirana je usporedba algoritama na problemima naprtnjače i n-kraljica, u sljedećim poglavljima.

Osnovni pojmovi iz područja kombinatoričkih i heurističkih algoritama se podrazumijevaju te uglavnom nisu definirani, eventualno su opisani u fusnotama. Implementacijski detalji također nisu opisivani već je naglasak stavljen na algoritme i rezultate testiranja (u prilogu rada nalazi se CD sa izvornim kodom).

Algoritmi su implementirani u programskom jeziku C++. Grafovi funkcija i drugi slikovni prikazi izrađeni su pomoću Matlab-a, programa za računanje u tehničkim znanostima.

[1] NP klasu problema predstavljaju problemi koji se ne mogu riješiti u polinomijalnom vremenu.
[2] Može se, ipak, primijetiti da je vrlo često konceptualno sasvim jednostavna heuristika vrlo uspješna.
[3] Genetski algoritam za rješavanje problema n-kraljica preuzet sa www.zemris.fer.hr/~golub/ga/ga.html a izvorni kod za simulirano kaljenje prilagođeno za problem naprtnjače preuzet sa www.math.mtu.edu/kreher/~cages.html

2. Heuristički algoritmi

2.1 Penjanje uzbrdo i mogućnosti njegove primjene

Penjanje uzbrdo (*hill climbing*) je heuristički algoritam temeljen na logici najstrmijeg uspona. Osnovna strategija ovog algoritma je da radi tako dugo dok heuristika generira rješenje koje je bolje od prethodnog. Kada se dogodi da novi kandidat nije bliži optimalnom od prethodnog, algoritam završava s radom.

Stoga je najzahtjevniji zadatak pri uporabi ovog algoritma, kao i kod ostalih, kreirati heuristiku[4], funkciju koja za trenutno rješenje daje još bolje. (Jedini izuzetak po ovom pitanju je genetski algoritam čija heuristika oponaša evolucijski proces, no i tu je puno slobode u prilagodbi algoritma problemu.)

Budući da je priroda heuristike takva da traži takvog boljeg kanidata koji je "sličan" ili "blizu" trenutnom rješenju, penjanje uzbrdo nema mehanizam za "bijeg" iz lokalnog optimuma. Unatoč tome, moguće je za mnoge probleme osmisliti uspješnu heuristiku; međutim vrlo često se ova metoda pokazuje previše ograničavajućom za nalaženje globalnog optimuma. U načelu, penjanje uzbrdo najuspješnije je kod problema koji nemaju lokalnih optimuma.

Glavna, pak, prednost ovog algoritma je što je konceptualno posve jednostavan, najjednostvniji među heurističkim algoritmima.

2.2 Simulirano kaljenje

2.2.1 Opis algoritma

Simulirano kaljenje *(simulated annealing)* je heuristički algoritam za globalno optimiranje čije ime i ideja potiču od postupka kaljenja u metalurgiji. Tehnikom kaljenja dobiva se materijal povećane čvrstoće i tvrdoće što se postiže reduciranjem defekata kristalne rešetke odnosno dovođenjem sustava u stanje manje unutarnje energije od početnog stanja.

Naime, u početku postupka, struktura materijala je u nekom lokalnom optimumu unutarnje energije. Zagrijavanjem sustava atomi napuštaju svoje prvobitne pozicije u kristalnoj rešetki te tijekom sporog hlađenja imaju šansu zauzeti neko drugo, povoljnije, mjesto i time sustav dovesti u stanje manje unutarnje energije nego je bila inicijalna (ili najmanje moguće – globalni optimum).

U algoritmu simulirano kaljenje svaki se element prostora rješenja može interpretirati kao stanje opisanog fizikalnog sustava a funkcija cilja kao unutarnja energija sustava u tom stanju. Algoritmu je cilj pronaći globalni optimum funkcije cilja kao što je u procesu kaljenja cilj pronaći konfiguraciju najmanje energije.

Kao što je kod kaljenja moguće da neka prijelazna konfiguracija ima veću unutarnju energiju od prethodne (a manju od inicijalne te veću od najmanje moguće) tako i simulirano kaljenje

[4] Heuristika je funkcija koja ima zadatak da u susjedstvu (dio prostora rješenje "sličan" ili "blizu" trenutnog rješenja) danog rješenja pronađe još boljeg kandidata, bilo iscrpnom pretragom, bilo nekom drugom metodom.

dopušta prijenos novog rješenja u sljedeću iteraciju čak i ako je lošije od prethodnog. Vjerojatnost za to je to manja što je temperatura manja. Dakle, kod prvih iteracija novo rješenje koje nije bolje od prethodnog s velikom se vjerojatnošću prenosi u sljedeću iteraciju (te se novi kandidat generira iz susjedstva tog rješenja). Kako se sustav hladi (broj iteracija približava maksimalnom broju) to je veća vjerojatnost da algoritam "ustraje" na trenutnom rješenju koje nije bolje od prethodnog.

Ova mogućnost da se lošiji kanidat ipak prenese u sljedeći ciklus omogućuje pretragu novih, prije neispitanih područja prostora rješenja te šansu da se naiđe na područje globalnog minimuma odnosno maksimuma.

2.2.2 Načini primjene

Prema tome, osnovni način "upravljanja" ovim algoritom biti će reguliranjem brzine hlađenja. Što je hlađenje sporije to je veće raspršenje kandidata te algoritam teži ka slučajnom pretraživanju prostora rješenja. Druga je krajnost premali koeficijent hlađenja – tada će vrlo vjerojatno algoritam pronaći lokalni optimum zbog preslabog raspršenja kanidata po prostoru rješenja.

Od tri ulazna parametra, broj iteracija, početna temperatura i koeficijent hlađenja, algoritam je najosjetljiviji na koeficijent hlađenja. Broj iteracija obično se kreće od 1000 do više 10-aka tisuća.

2.3 Tabu pretraživanje i mogućnosti njegove primjene

Osnovna je ideja kod heuristika da već male promjene mogu unaprijediti trenutno rješenje. Tako je u primjeni vrlo raširena heurisitka koja koristi binarni vektor za prikaz elemenata prostora rješenja te koja susjedstvom trenutnog rješenja smatra sve vektore koji se od danog razlikuju u jednome bitu (Hammingova udaljenost jednaka je 1).

Posebnost tabu pretraživanja *(tabu search)* među heurističkim algoritmima je što heuristika vraća najboljeg kandidata iz susjedstva trenutnog rješenja. Dakle, u novi ciklus se prenosi onaj element susjedstva trenutnog rješenja za kojeg funkcija cilja ima najveću ili najmanju vrijednost – ovisno o smjeru optimizacije. Često se za nalaženje najboljeg kandidata koristi iscrpna pretraga.

Za razliku od penjanja uzbrdo i simuliranog kaljenja, tabu pretraživanje u sljedeći ciklus uvijek prenosti novokreiranog kandidata, i u slučaju ako je bolji od trenutnog rješenja i u slučaju ako to nije. Na taj se način pretražuju nova područja prostora rješenja, dok heuristika ima zadatak da u danom području pronađe najboljeg kandidata.

Budući da heuristika kad ovog algoritma pronalazi najboljeg kandidata u susjedstvu, neracionalno bi bilo više puta generirati istog kandidata. Kako bi se to izbjeglo, formira se "tabu lista" u koju se na neki način zapisuju kandidati čije je susjedstvo već pretraženo. Tako se u pojedinoj iteraciji ne pretražuje cijelo susjedsvo već samo oni njegovi elementi koji nisu zabilježeni u "tabu listi". Obično se ne zapisuju sami kandidati već funkcije kojima su oni generirani.

2.4 Genetski algoritam i mogućnosti njegove primjene

Genetski algoritam[5] *(genetic algorithm)* je heuristička metoda rješavanja problema, obično optimiranja, koji oponaša prirodni evolucijski proces. Tako, prostor rješenja kod genetskog algoritma čine jedinke, predstavljene kromosomima[6]. Jedinke su prepuštene procesu evolucije, koji u sljedeću generaciju prenosi one s najboljim svojstvima tj. one za koje funkcija cilja ima najveće vrijednosti. Trajanje evolucije unaprijed zadajemo (na više načina; primjerice brojem iteracija). Osnovni mehanizmi odabira sve boljih i boljih jedinki (kromosoma, tj. kanidata za rješenje), su kao i kod prirodnog procesa evolucije selekcija, križanje i mutacija.

Točnije, selekcijom odabiremo određeni broj najboljih kromosoma za sljedeću generaciju (iteraciju), koji će križanjem formirati nove kromosome s ciljem dobivanja još boljih kandidata. Nakon toga, djeluje genetski opetator mutiranja, čiji je zadatak formiranje kanditata iz neistraženog prostora rješenja (obično se pritom čuva elitizam, tj. trenutno najbolje rješenje je zaštićeno od mutacije). Dakle, dok je križanje zaduženo za konvergenciju prema optimumu (budući da je velika vjerojatnost da se križanjem dobrih kandidata dođe do još boljih, a sličnih), mutacija omogućava stalnu potragu cijelog prostora rješenja i time bijeg iz lokalnog optimuma, tj. pronalaženje kandidata u području globalnog optimuma.

Genetski algoritmi su se pokazali vrlo efikasnima u rješavanju raznih problema iz inžinjerske prakse, bilo da se radi o generiranju kombinatoričkih struktura ili opcimizacijskoj klasi problema. Opće osobine genetskog algoritma jesu robusnost, primjenjivost na široki spektar problema te pouzdanost (u smislu da nađeno rješenje, s velikom vjerojatnošću, ne odstupa puno od optimalnog). Za razliku od ostalih heurističkih algoritama, u svakoj iteraciji barataju sa skupom kadnidata za rješenje (populacijom) a ne jednim kandidatom.

Jedna od teškoća u primjeni genetskog algoritma je veliki uticaj ulaznih parametara na efikasnost. Kao i kod ostalih heurističkih algoritama, jedini način optimiranja parametara je eksperimentiranjem. Ipak, neke vrijednosti parametara su se pokazale "standardnima" pa mogu poslužiti kao polazna točka za traženje optimalnih parametara danog problema (tablica 2.1) (Golub, 2004a).

Parametri	Oznaka	*"Mala"* populacija	*"Velika"* populacija
Veličina populacije	*VEL_POP*	30	100
Vjerojatnost mutacije	p_m	0.01	0.001
Vjerojatnost križanja	p_c	0.9	0.6
Broj jedinki za eliminaciju	*M*	*VEL_POP*/2	*VEL_POP*/4

Tablica 2.1 *Parametri genetskog algoritma s generacijskom i eliminacijskom reprodukcijom*

[5] Genetski algoritmi predloženi su od strane J. H. Hollanda ranih 70-ih godina 20. st.
[6] Kada govorimo o genetskim algoritmima, jedinka i kromosom su sinomimi – budući da kromosom sadrži sve informacije o jedinki.

3. Optimiranje realnih funkcija jedne varijable

3.1 Prikaz realnog broja pomoću binarnog vektora

Kod uspoređivanih heurističkih algoritama, osim kod penjanja uzbrdo, koristi se binarni vektor za prikaz realnog broja, koji predstavlja kodiranu vrijednost $x \in [dg, gg]$.

Vektor čiji su svi bitovi postavljeni na nulu predstavlja donju granicu intervala a vektor sa svim jedinicama gornju granicu. Točna pretvorba kromosoma u realni broj izvodi se prema relaciji

$$x = dg + \frac{b}{\left(2^n - 1\right)}\left(gg - dg\right),$$

pri čemu je *dg* donja granica intervala, *gg* gornja granica a *b* broj kojeg predstavlja kromosom. Obrnuto, pretvorba realnog broja *x* u kromosom – binarni zapis izvodi se prema relaciji:

$$b = \frac{x - dg}{gg - dg}\left(2^n - 1\right).$$

Broj bitova izračunava se prema relaciji

$$n \geq \frac{\log\left[(gg - dg) * 10^p + 1\right]}{\log 2}.$$

3.2 Optimiziranje funkcija primjenom heurističkog algoritma penjanje uzbrdo

3.2.1 Opis heuristike

```
Heuristika (xsd, xsg) {
    razdijeli interval [xsd, xsg] na n ekvidistantnih dijelova
    početak svakog intervala i kraj zadnjeg su kandidati
    odaberi najboljeg kadidata i zapamti njegove susjede
    return (najbolji kandidat)
}

Algoritam Penjanje_uzbrdo_OptFje(n){
     Y=Heuristika (dg, gg);
     Xbest=Y;
     searching=true;
     while (searching) {
          Y=Heuristika (xsd, xsg);
          if Profit(Y) > Profit(Xbest) then Xbest=Y;
          else searching=false;
     }
     return (Xbest);
}
```

Algoritam 3.1 *Heuristički algoritam penjanje uzbrdo primijenjen na optimiranje funkcija*

Budući da heuristički algoritam penjanje uzbrdo završava s radom ukoliko novi kandidat nije bolji od prethodnog, važno je da heuristika bude efikasna. Za problem optimiranja funkcija osmišljena je vrlo jednostavna heuristka koja, u prvom koraku, interval na kojem se traži optimum funkcije podijeli na određeni broj (n) ekvidistantnih intervala, čije granice predstavljaju susjedstvo trenutnog rješenja (algoritam 3.1). Najbolji kandidat postaje novo rješenje a njegovi susjedi postaju granice intervala na kojem se traži sljedeće bolje rješenje. Broj podjela intervala n je ulazni parametar algoritma.

Nedostatak pristupa je mogućnost "preskakanja" globalnog optimuma ako je podjela danog intervala pregruba. To se, dakako, može popraviti preciznijom podjelom intervala, no tada se povećava broj izračunavanja vrijednosti funkcije cilja. Što funkcija ima manji omjer širine globalnog optimuma i prostora rješenja, to će podjela trebati biti preciznija.

3.2.2 Rezultati testiranja

Rezultate testiranja prikazuje tablica 3.1. Prva tri stupca prikazuju testiranu funkciju, interval na kojem se tražio optimum i vrijednost optimuma na 6 decimala točnosti. U prvom testu pokrenut je algoritam sa 10 podjela intervala ($n=10$); rezultate prikazuje 4. stupac tablice. Nakon toga, postavilo se pitanje za koliki n (u koracima 50, 100, …) će optimum biti precizno određen; rezultate tog, drugog, testa prikazuje zadnji stupac tablice.

Funkcija	Interval	f_{opt}	$n=10$		$\lvert f(x_{best}) - f_{opt}\rvert=0$	
			$\lvert f(x_{best}) - f_{opt}\rvert$	numiter	n	numiter
$f_{p0}(x) = x^3 - 4x$	[-2, 2]	3.079201	0	6		
$f_{p1}(x) = \frac{1}{4}x^4 - x^3 - 2x^2 + 1$	[-2, 5]	-31	0	11		
$f_{p2}(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$	[-1, 1]	-0.428571	0	8		
$f_0(x) = 1 + x\sin(10\pi x)$	[-1, 2]	2.850275	0.199967	10	100	5
$f_1(x) = \frac{\sin(10\pi x^2)}{x}$	[-1, 2]	4.771197	0.015916	1	50	6
$f_2(x) = x^2\sin(10\pi x^2)$	[1, 2]	3.850135	0.399985	7	50	8
$f_3(x) = x^{\sin(x)}$	[0, 50]	45.555967	0.000117	4	100	7
$f_4(x) = x\cos[tg(x)]$	[-2, 6]	5.747734	0	2		
$f_5(x) = x\cos(tg\sqrt{x})$	[0, 5]	2.989384	0.016944	1	50	7
$f_6(x) = x\sin\left\{\frac{30}{x}\left[\frac{\pi}{2} - arctg(x)\right]\right\}$	[-1, 8]	4.632533	0	4		
$f_7(x) = 0.5 - \frac{\sin^2(x) - 0.5}{(1 + 0.001x^2)^2}$	[-100, 100]	1	0	1		

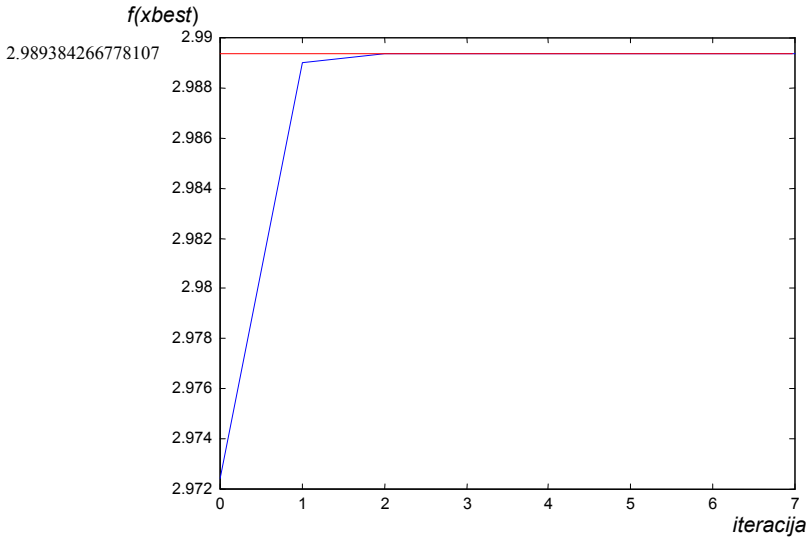
Tablica 3.1 *Rezultati optimiranja funkcija penjanjem uzbrdo*

Odmah se uočava da je algoritam vrlo osjetljiv na oblik funkcije. Za prve tri, polinomske funkcije koje imaju jedan ili dva lokalna maksimuma, kao i za ostale slične, algoritam je bio vrlo efikasan. Pronašao je optimalno rješenje na traženu točnost u svega nekoliko iteracija. Može se zaključiti da što je veća gustoća lokalnih optimuma na prostoru rješenja odnosno što

je njihova vrijednost bliža vrijednosti globalnog optimuma to algoritam teže nalazi točno rješenje (odnosno *n* treba biti veći da bi algoritam bio uspješan). Može se još reći da *n* treba biti to veći što je omjer širine prostora na kojem se nalazi globalni maksimum i širine prostora rješenja manji, i što je vrijednost lokalnih optimuma bliža vrijednsti globalnog optimuma.

Druga karakteristika algoritma koja se uočava je brzina algoritma tj. mali broj potrebnih izračunavanja vrijednosti funkcije cilja. U najgorem slučaju (kod funkcije f_3) vrijednost funkcije cilja se izračunala 700 puta, a za usporedbu može se navesti da se kod genetskog algoritma s populacijom od 50 i vremenom evolucije od 100 iteracija profit računa 5000 puta (uz izvođenje ostalih operacija). Slika 3.1 prikazuje taj brzi napredak prema optimalnom rješenju tijekom iteracija.

Za primijetiti je da je u prvom testu najveće odstupanje od optimuma bilo kod funkcije f_2, koja će se kasnije pokazati za sve algoritme najzahtjevnijom.



Slika 3.1 *Poboljšavanje rješenja tijekom iteracija za funkciju f_5*

3.3 Optimiranje funkcija primjenom simuliranog kaljenja

3.3.1 Opis heuristike

Odabrana je posve jednostavna heuristika: susjedstvo danog vektora čine vektori s Hammingovom udaljenošću jednakom 1. Za dani vektor (koji predstavlja realni broj-kandidat za rješenje) heuristika vraća vektor koji se od ulaznog razlikuje u jednom, slučajno odabranom, bitu (algoritam 3.2).

```
Algoritam Simulirano_kaljenje_OptFje(c_max, T_0, α){
    T= T_0
    X=[X_0, … X_n-1]
    X_best =X
    for c=0 to cmax {
        j=Random(0, n-1) //"klasična" heuristika
        Y=X, Yj=1-Xj
        if P(Y) > P(X) then {
            if P(Y) > P(X_best) then X_best =Y
            X=Y
        }
        else
            r=Random(0-1)
            if r<e^((P(y)-P(x))/T) then X=Y
        T=αT_0
    }
    return (xbest)
}
```

Algoritam 3.2 *Za optimiranje funkcija korišteno je simulirano kaljenje s heuristikom koja mijenja slučajno odabrani bit*

3.3.2 Rezultati testiranja

Unatoč ovako jednostavnoj heuristici, rezultati su vrlo dobri, kako prikazuje tablica 3.2.

Funkcija	Interval	f_{opt}	c_{max}=1000, α=0.98		c_{max}=2000, α=0.985	
			Broj nađenih optimuma u 10 pokretanja	Odstupanje najlošijeg rješenja od optimuma	Broj nađenih optimuma u 10 pokretanja	Odstupanje najlošijeg rješenja od optimuma
f_{p0}	[-2, 2]	3.079201	5	0.0017 %	6	0.3 %
f_{p1}	[-2, 5]	-31	2	0.0049 %	10	
f_{p2}	[-1, 1]	-0.428571	6	0.0512 %	7	0.001 %
f_0	[-1, 2]	2.850275	3	10.4374 %	3	8.9 %
f_1	[-1, 2]	4.771197	5	6.3026 %	4	4.8 %
f_2	[1, 2]	3.850135	0	25.9719 %	1	14.5 %
f_3	[0, 50]	45.555967	2	10.8 %	2	10.8 %
f_4	[-2, 6]	5.747734	7	9.512 %	8	35.7 %
f_5	[0, 5]	2.989384	2	11.7973 %	2	12.4 %
f_6	[-1, 8]	4.632533	4	0.0727 %	6	0.01 %
f_7	[-100, 100]	1	7	0.9895 %	6	0.9899 %

Tablica 3.2 *Rezultati optimiranja funkcija simuliranim kaljenjem*

3.4 Optimiranje funkcija primjenom tabu pretraživanja

3.4.1 Opis heuristike

Za problem optimiranja realnih funkcija jedne varijable najprije je primijenjen osnovni, konceptualno posve jednostavni, algoritam tabu pretraživanja. Za zapis realnog broja koristi se binarni vektor a susjedstvo predstavljaju svi vektori koji se od danog razlikuju u jednom bitu (kao i kod simuliranog kaljenja).

Tabu lista je osmišljena tako da se u nju zapisuje redni broj promijenjenog bita. Na taj način heuristika neće ponovo razmatrati već generirane kandidate; točnije, za to je manja vjerojatnost (to manja što je duljina tablu liste veća).

Nakon prvog testa optimiranja, rezultati, koje prikazuje tablica 3.2 nisu bili zadovoljavajući.

Funkcija	Interval	f_{opt}	Broj nađenih optimuma u 10 pokretanja
f_{p0}	[-2, 2]	3.079201	2
f_{p1}	[-2, 5]	-31	0
f_{p2}	[-1, 1]	-0.428571	3
f_0	[-1, 2]	2.850275	0
f_1	[-1, 2]	4.771197	0
f_2	[1, 2]	3.850135	0
f_3	[0, 50]	45.555967	4
f_4	[-2, 6]	5.747734	2
f_5	[0, 5]	2.989384	0
f_6	[-1, 8]	4.632533	3
f_7	[-100, 100]	1	6

Tablica 3.3 *Rezultati sa prvom heuristikom,* c_{max}=1000*,* L=10

Osim male vjerojatnosti nalaženja globalnog optimuma na zadanu točnost, odstupanja od otpimuma su bila velika (veća od karakterističnih za heursitičke algoritme - cca 5%). Takvi rezultati ukazivali su da algoritam ima preslab mehanizam za raspršivanje kandidata po prostoru rješenja.

Stoga je poboljšan algoritam tako da je generiran slučajni vektor svaki puta kad novoformirani kandidat nije predstavljao bolje rješenje od prethodnog.

3.4.2 Rezultati testiranja

Rezultate testiranja s poboljšanim algoritmom prikazuje tablica 3.4.

Funkcija	Interval	f_{opt}	c_{max}=500, L=1		c_{max}=1000, L=1	
			Broj nađenih optimuma	Odstupanje najlošijeg rješenja	Broj nađenih optimuma	Odstupanje najlošijeg rješenja
f_{p0}	[-2, 2]	3.079201	9	0.0003%	10	
f_{p1}	[-2, 5]	-31	8	0.0001%	10	
f_{p2}	[-1, 1]	-0.428571	10		10	
f_0	[-1, 2]	2.850275	6	<0.000001%	10	
f_1	[-1, 2]	4.771197	6	0.0003%	9	0.0001%
f_2	[1, 2]	3.850135	2	0.002%	6	0.002%
f_3	[0, 50]	45.555967	6	0.0009%	6	0.0009%
f_4	[-2, 6]	5.747734	10		10	
f_5	[0, 5]	2.989384	4	0.0001%	6	0.0001%
f_6	[-1, 8]	4.632533	10		10	
f_7	[-100, 100]	1	10		10	

Tablica 3.4 *Rezultati optimiranja funkcija tabu pretraživanjem*

Tabu pretraživanje se pokazalo vrlo pouzdanim algoritmom za optimiranje funkcija. U većini slučaja generirano je točno rješenje na traženu preciznost ili je odstupanje bilo vrlo malo. Sumarne rezultate prikazuje sljedeća tablica, 3.5

Parametri algoritma	c_{max}=500, L=1	c_{max}=1000, L=1
Vjerojatnost da algoritam nađe globalni optimum	74%	88%
Očekivano odstupanje najlošijeg rješenja	0.002%	0.002%
Broj izračuna fje cilja	cca 10000	cca 20000

Tablica 3.5 *Pouzdanost i efikasnost tabu pretraživanja kod optimiranja realnih funkcija jedne varijable*

3.5 Optimiranje funkcija primjenom genetskog algoritma

3.5.1 Uvod

Dosezi i mogućnosti genetskog algoritma testirani su na problemu optimizacije funkcija. Odabrano je 11 različitih realnih funkcija jedne varijable, od polinoma do zahtjevnih funkcija s mnoštvom lokalnih optimuma. Ispitne funkcije odabrane su s nastojanjem da budu reprezentativne, kako bi se, eventualno, došlo do nekih generalizacija o ponašanju genetskog algoritma na ovoj klasi problema.

Konkretno, cilj je ustanoviti efikasnost genetskog algoritma na ovoj klasi problema. Što je broj generacija (iteracija) potrebnih da se dođe do globalnog opitimuma, odnosno broj izračuna funkcije cilja, manji to je algoritam efikasniji. Karakteristika heurističkih algoritama je da, u pravilu, daju ili optimalno rješenje ili rješenje koje je blisko opitmalnom; međutim za to nema nikakve čvrste garancije. Načelno, za realni problem koji se rješava heurističkim algoritmom ne može se znati koliko je algoritam pouzdan, tj. koliko se dobiveno rješenje razlikuje od optimalnog. Međutim, određena sigurnost ipak postoji zbog statističke zakonitosti; u većini slučajeva pouzdanost algoritma je velika pa očekujemo da će tako biti i za rješavani problem. Posebice će to vrijediti ako je primijenjeno na određenu klasu problema. Zato će se u testiranjima ustanoviti koliko se prosječno rješenje razlikuje od optimalnog kako bi se utvrdila očekivana pouzdanost za cijelu klasu problema. Treća mjerena karakteristika algoritma je preciznost.

Na osnovi dobivenih rezultata nastojati će se "standardizirati" ulazni parametri genetskog algoritma kada se on primjenjuje na optimiranje realnih funkcija jedne varijable. Promatrati će se da li se uopće može utvrditi neka zakonitost, da li se parametri podudaraju sa općim parametrima, predstavljenim u tablici 2.1, ili postoje neke specifičnosti.

3.5.2 Genetski algoritmi s eliminacijskom selekcijom

```
Genetski_algoritam {
    Generiraj slučajno VEL_POP jedinki;
    Sve dok (ima_vremena) radi
        Evaluiraj i selektiraj;
         - izračunaj vrijednosti fje cilja za sve jedinke
         - primijeni translaciju
         - izračunaj vrijednosti fje kazne za sve jedinke
         - izbriši M jedinki na način da je vjerojatnost brisanja prop. vrijednosti fje kazne
        Križaj s jednom točkom prekida;  //kod GA_2 UniformnoKrižaj;
         - s jednakom vjerojatnošću odaberi roditelje od opstalih jedinki
         - ako su jedinke jednake  tada mutiraj jednu od njih
                inače primijeni križanje s jednom točkom prekida
          - ponavljaj postupak sve dok se ne generira M djece
        Jednostavno mutiraj;
         - radi za svaku jedinku u populaciji, samo ne najbolju:
            - radi za svaki bit tekuće jedinke:
                - izračunaj vjerojatnost i po potrebi promijeni bit
     Ispiši zadnju generaciju;
}
```

Algoritam 3.3 *Jednostavni eleminacijski genetski algoritam*

Razvijena su dva genetska algoritma, oba s eliminacijskom selekcijom i jednostavnom mutacijom koja čuva najbolje rješenje (algoritam 3.3). Kod oba algoritma prilikom križanja mutira jedan od roditelja ako se uspostavi da su oba kromosoma – roditelja jednaka. Jedina razlika među algoritmima je što jedan algoritam (GA_1) koristi križanje s jednom točkom prekida a drugi (GA_2) uniformnmo križanje.

Algoritmi su pokrenuti sa sljedećim parametrima (tablica 3.6):

Algoritam	GA_1	GA_2
Vrsta GA	eliminacijski, M=50%	…
Veličina populacije	50	…
Prikaz	binarni, broj bitova=*f(dg,gg,p)*	…
Ispravak funkcije dobrote	translacija	…
Eliminacija duplikata	da, prilikom križanja	…
Križanje	s jednom točkom prekida, p_c=0.5	uniformno
Mutacija	jednostavna	…

Tablica 3.6 *Tip i parametri ugrađenog genetskog algoritma*

Dakle, algoritmi koriste binarni prikaz kromosoma čija duljina (broj bitova) ovisi o granicama intervala i preciznosti. Ukoliko su roditelji jednaki, mutira jedan od njih. Kod prvog algoritma križanje je jednostavno, s jednom točkom prekida, a kod GA_2 križanje je uniformno. Populaciju je, ovisno o testu, činilo 50 ili 100 jedinki a broj kromosoma za eliminaciju je uvijek iznosio 50% veličine populacije.

3.5.3 Rezultati testiranja

Napravljene su dvije serije testova, čije rezulate prikazuju tablice 3.7 i 3.8. U prvoj seriji, cilj je bio maksimizirati efikasnost algoritama tj. pokrenuti algoritme sa što manjim brojem iteracija a da rezultati budu prihvaljivi. U drugoj seriji ispitivao se utjecaj povećanja populacije na rezultat.

Funkcija	Interval	f_{opt}	Broj nađenih optimuma u 10 pokretanja	
			GA_1 (križanje s jednom točkom pr.)	GA_2 (uniformno križanje)
f_{p0}	[-2, 2]	3.079201	4	7
f_{p1}	[-2, 5]	-31	5	8
f_{p2}	[-1, 1]	-0.428571	6	7
f_0	[-1, 2]	2.850275	4	4
f_1	[-1, 2]	4.771197	6	5
f_2	[1, 2]	3.850135	2	4
f_3	[0, 50]	45.555967	6	8
f_4	[-2, 6]	5.747734	10	10
f_5	[0, 5]	2.989384	6	3
f_6	[-1, 8]	4.632533	7	8
f_7	[-100, 100]	1	7	6

Tablica 3.7 *Genetski algoritmi pokrenuti s parametrima VEL_POP*=50*, t*=100*, M*=50% *i* p_m=0.01

Primijećen je veći broj nađenih optimalnih rješenja kod drugog algoritma, s uniformnim križanjem. Dok je kod GA_1 ukupno 63 nađenih optimuma, kod GA_2 taj broj iznosi 70.

Rijetko se koji rezultat poboljšao povećanjem broja iteracija sa 100 na 300, a sa 300 na 1000 niti jedan (osim ako je ujedno bila povećana vjerojatnost mutacije). Pokazalo se da je algoritam s ovim ulaznim parametrima najsenzitivniji na vjerojatnost mutacije.

U drugoj seriji testova ispitivano je kako pouzdanost i preciznost algoritma ovise o veličini početne populacije. Budući da je veličina populacije sada povećana, smanjena je stopa mutacije, kako sugeriraju drugi radovi (Golub, 2004a).

Funkcija	Interval	f_{opt}	Broj nađenih optimuma u 10 pokretanja	
			GA_1 (križanje s jednom točkom prekida)	GA_2 (uniformno križanje)
f_{p0}	[-2, 2]	3.079201	8	9
f_{p1}	[-2, 5]	-31	7	9
f_{p2}	[-1, 1]	-0.428571	9	10
f_0	[-1, 2]	2.850275	6	6
f_1	[-1, 2]	4.771197	6	9
f_2	[1, 2]	3.850135	4	8
f_3	[0, 50]	45.555967	8	9
f_4	[-2, 6]	5.747734	10	10
f_5	[0, 5]	2.989384	5	9
f_6	[-1, 8]	4.632533	8	9
f_7	[-100, 100]	1	8	10

Tablica 3.8 *Genetski algoritmi pokrenuti s parametrima* VEL_POP=100, t=1000, M=50% *i* p_m=0.007

3.5.4 Optimalni parametri jednostavnog genetskog algoritma s eliminacijskom selekcijom

Primijenjeni algoritmi (s križanjem s jednom točkom prekida i s uniformnim križanjem) uspješni su u optimiranju funkcija te pokazuju karakteristike koje su u skladu s dosadašnjim rezultatima u primjeni genetskih algoritama. Tako su u provedenim testovima algoritmi bili najosjetljivi na stopu mutacije. Također se pokazalo da je algoritam s uniformnim križanjem bio uspješniji od algoritma s križanjem s jednom točkom prekida, generiravši više optimalnih rješenja.

Najprije su algoritmi testirani u stanju velike efikasnosti (tako da broj iteracija bude što manji -100) (tablica 3.7). S dotičnim su parametrima algoritmi pokazivali slabo povećanje preciznosti i pozdanosti s povećanjem broja iteracija. Algoritmi su bili najsenzitivniji na stupu mutacije te je variranjem iste od 0.01 do 0.03 (ili još nešto više) posve precizno, na traženih 6 decimala, određen optimum svake ispitne funkcije u samo jednom pokretanju algoritma.

Druga serija testova, u kojoj je povećana veličina populacije (s ciljem da se ispita ovisnost pouzdanosti i preciznosti o veličini populacije) pokazuje da povećanje broja iteracija značajno doprinosi boljem rješenju ukoliko je populacija veća (a vjerojatnost mutacije manja).
Dakle, općenito se može reći da kada se jednostavni genetski algoritam s eliminacijskom selekcijom, a isto tako i opisani genetski algoritam s uniformnim križanjem, primjenjuje za optimiranje realnih funkcija jedne varijable, s većom populacijom (100), većim brojem iteracija (1000) i manjom stopom mutacije (0.007) dobivamo precizniji i pouzdaniji rezultat.

3.6 Usporedba heurističkih algoritama na primjeru optimiranja realne funkcije jedne varijable

3.6.1 Osnovne karakteristike uspoređivanih algoritama

Tablica 3.11 prikazuje nam osnovne karakteristike uspoređivanih algoritama.

Heuristički algoritam	Penjanje uzbrdo	Simulirano kaljenje	Tabu pretraživanje	Genetski algoritam
Zapis elemenata prostora rješenja	Broj s pomičnom točkom	Binarni vektor	Binarni vektor	Binarni vektor
Heuristika	Susjedstvo čini n brojeva između prvog donjeg i prvog gornjeg susjeda trenutnog rješenja	Susjedstvo čine vektori koji se od danog razlikuju u jednom bitu	Susjedstvo čine vektori koji se od danog razlikuju u jednom bitu a nisu u tabu listi	Susjedstvo čini populacija koja se od dane formira križanjem
	Vraća najboljeg od ispitanih kandidata i pamti njegove susjede	Vraća slučajno generiranog kandidata	Vraća najboljeg mogućeg kandidata (iscrpna pretraga)	Vraća novu generaciju te odabire ponajboje kandidate
Mehanizam konvergencije	U svakoj iteraciji interval na kojem se traži rješenje je sve uži	Novoformirani kandidat se vrlo često ne razlikuje puno od prethodnog	Iscrpna pretraga susjedstva trenutnog kandidata	Križanje
Mehanizam raspršenja	Kandidati su ekvidistantno raspoređeni po intervalu	I lošiji kandidat ima šansu (to veću što je temp. veća) ići u sljedeću iteraciju	Zamjena novog, lošijeg kandidata sluč. generiranim vektorom	Mutacija
Ulazni parametri	- broj razmatranih kandidata na intervalu (n)	- broj iteracija (c_{max}) - početna temperatura (T_0) - koeficijent hlađenja (α)	- broj iteracija (c_{max}) - kapacitet tabu liste (L)	- broj iteracija (t) - veličina populacije (VEL_POP) - postotak selekcije (M) - vjerojatnost mutacije (p_m)

Tablica 3.11 *Osnovne karakteristike uspoređivanih heurističkih algoritama*

3.6.2 Usporedni rezultati testiranja i zaključak

Heuristički algoritam		HC	SA		TS		GA_1		GA_2	
		test 1	test 1	test 2	test 1	test 2	test 1	test 2	test 1	test 2
Ulazni parametri	numiter		1000	2000	500	1000	300	1000	300	1000
		n=100	c_{max}=1000 α=0.98	c_{max}=2000 α=0.985	L=1	L=1	*VEL_POP*=50 p_m=0.03 M=50	*VEL_POP*=100 p_m=0.007 M=50	*VEL_POP*=50 p_m=0.03 M=50	*VEL_POP*=100 p_m=0.007 M=50
PRECIZNOST	broj decimala	15	6	6	6	6	6	6	6	6
POUZDANOST	vjerojatnost optimalnog rješenja[7]	100%	39%	50%	74%	88%	57%	71%	63%	89%
	odstupanje najlošijeg rješenja od optimuma[8]		26%	35%	0.002%	0.002%	5%	5%	5%	4.7%
EFIKASNOST	broj izračuna vrijednosti funkcije cilja	100-700	1000	2000	10000	20000	15000	100000	15000	100000
Osjetljivost na ulazne parametre			na c_{max} gotovo neosjetljiv na α vrlo osjetljiv		na $L \neq 1$ gotovo neosjetljiv		vrlo osjetljiv na p_m veća populacija sa većim t doprinosi pouzdanosti		vrlo osjetljiv na p_m veća populacija sa većim t doprinosi pouzdanosti	

Tablica 3.12 *Usporedni test heurističkih algoritama penjanja uzbrdo, simuliranog kaljenja, tabu pretraživanja i genetskog algoritma na primjeru optimiranja realnih funkcija jedne varijable*

[7] Na osnovi 10 testova za pojedinu funkciju
[8] Kod svakog algoritma najveće je ostupanje od optimuma zabilježeno kod funkcije f_2, osim u jednom testu sa simuliranim kaljenjem

Tablica 3.12 prikazuje uspoređene rezultate optimiranja realnih funkcija jedne varijable heurističkim algoritmima penjanje uzbrdo, simulirano kaljenje, tabu pretraživanje i genetskim algoritmima. Algoritmi su primijenjeni na uzorku od 11 funkcija, odabranih s nastojanjem da budu reprezentativne za ovu klasu problema.

Svi su se algoritmi pokazali uspješnima u rješavanju zadanog problema, došavši do rješenja tražene točnosti u većini pokretanja. Kao najefikasniji, najpreciziji i najpouzdaniji pokazao se algoritam penjanje uzbrdo, koji je ujedno i konceptualno najjednostavniji među analiziranim algoritmima. Sa zadanim parametrom $n=100$ u svakom pokretanju, za svaku funkciju, dobiveno je rješenje na 15 decimala točnosti.

Općenito, može se primijetiti da povećanje broja iteracija ne dovodi uvijek do boljeg rješenja. Primjerice, kod genetskog algoritma nakon nekog određenog broja ($t=300$) dalnjim povećanjem iteracija za dane parametre ne dobivamo bolje rješenje. Tek ako se uz povećanje broja iteracija poveća i populacija dobivaju se stabilniji rezultati. Druga izražena karakteristika svih algoritama je osjetljivost na tip funkcije. Tako se praktički u svim testovima najzahtjevnijom pokazala funkcija f_2.

4. Usporedba heurističkih algoritama na problemu naprtnjače

U radu se uspoređuju heuristički algoritmi simulirano kaljenje, tabu pretraživanje i genetski algoritam na primjeru problema naprtnjače. Prikazuju se heuristike kao i rezultati testiranja za pojedini algoritam. Izvedeni su zaključci o primjeni heurističkih algoritama na ovoj NP klasi problema, kao i njihova usporedba.

4.1 Problem naprtnjače

Problem naprtnjače je kombinatorički problem u kojem je potrebno maksimizirati funkciju cilja. Dano je *n* grumena koji sadrže "zlato" i naprtnjača kapaciteta *W*. U torbu treba spremiti onu kombinaciju grumena koja je najvrednija tj. koja sadrži najviše "zlata" (slika 4.1).

Dano je:	profiti	$p_0, p_1, \ldots p_{n-1}$
	težine	$\omega_0, \omega_1, \ldots \omega_{n-1}$
	kapacitet	W
Treba naći:	n-torku $[x_0, \ldots x_{n-1}] \in \{0,1\}^n$ takvu da	
	$P = \sum_{i=0}^{n-1} p_i x_i \rightarrow \max$, i	
	$\sum_{i=0}^{n-1} w_i x_i \leq W.$	

Slika 4.1 *Problem naprtnjače*

Problem naprtnjače je NP težak problem; budući da za *n* grumena prostor rješenja broji 2^n vektora. Tablica 4.7 prikazuje veličinu prostora rješenja za nekoliko dimenzija problema.

Iako i među determinstičkim tehnikama ima onih koje reduciraju prostor rješenja, one nisu praktične za $n > (\approx 40)$ (Kreher, Stinson, 1999).

Inače, problem naprtnjače ima važnu primjenu u kriptografiji, poslovnom sektoru i drugdje.

4.2 Opis heuristika

Kod simuliranog kaljenja korištena je konceptualno jednostavna heuristika, ista kao i kod optimiranja funkcija: za dano rješenje heuristika vraća vektor koji se od danog razlikuje u jednom, slučajno odabranom bitu.

Heuristika kod tabu pretraživanja je konceptualno složenija; ponaša se prema algoritmu 4.1. Ideja je te heuristike da se naprtnjača stalno "puni i prazni", na način da se u nju doda grumen s najvećim relativnim doprinosom profita a kada to nije moguće onda se oduzme grumen s najmanjim omjerom profita i težine.

Algoritam TS_Heuristika_Naprtanjaca (x[])

1. ako postoji barem jedan *i* za koji vrijedi
   $x_i = 0$ i *i* nije u tabu listi tada

   među takvim *i*-evima nađi onaj za kojeg je $\frac{p_i}{w_i} \rightarrow \max$

   te promijeni pripadni bit u 1

2. inače razmatraj i-eve za koje $x_i = 1$

   među njima nađi onaj za kojeg je $\frac{p_i}{w_i} \rightarrow \min$

   te promijeni pripadni bit u 0

Algoritam 4.1 *Heuristika tabu pretraživanja za problem naprtnjače*

Korištene su dvije inačice genetskog algoritma, koje se razlikuju jedino u vrsti križanja. Uspoređen je genetski algoritam s eliminacijskom selekcijom, jednostavnom mutacijom i križanjem s jednom točkom prekida (GA_1), te genetski algoritam koji se od prethodnog razlikuje jedino po uniformnom križanju (GA_2).

4.3 Rezultati testiranja

Za testne primjere su odabrani primjeri s dimenzijama od 15, 25, 50 i 100. U prilogu B su prikazani dotični primjeri problema naprtnjače. Za primjere sa 15 i 25 "grumena" rješenje je provjereno deterministčkim metodama. Slijedi opis rješavanja problema dimenzija 15 i 25, a analogno vrijedi i za ostala dva primjera. Efikasnost algoritama uspoređena je na sva 4 primjera, a sumarne rezultate prikazuje tablica 4.7.

α	c_{max}	Min	Max	Prosjek	Broj nađenih optimuma u 10 pokretanja
0.999	1000	1019	1051	1035.6	0
	5000	1024	1053	1036.2	2
	20000	1030	1053	1044.6	2
0.9995	1000	1005	1053	1033.0	2
	5000	1037	1053	1047.7	2
	20000	1046	1053	1051.4	6
0.9999	1000	1019	1051	1033.0	0
	5000	1040	1053	1049.4	2
	20000	1048	1053	1052.5	9

Tablica 4.1 *Rezultati rješavanja problema naprtnjače sa* 15 *grumena simuliranim kaljenjem*

α	c_{max}	Min	Max	Prosjek	Broj nađenih optimuma u 10 pokretanja
0.999	1000	6367	7263	6922.4	0
	5000	6789	7349	7169	0
	20000	7022	7317	7188.1	0
0.9995	5000	7163	7419	7261.8	0
	20000	7148	7464	7281.8	1
	100000	7163	7464	7320.1	1
0.9999	20000	7327	7464	7413.2	1
	100000	7365	7464	7422.5	3
	500000	7321	7464	7420.0	3
0.99999	500000	7464	7464	7464.0	10

Tablica 4.2 *Rezultati rješavanja problema naprtnjače sa* 25 *grumena dobiveni simuliranim kaljenjem*

Kod rješavanja problema naprtnjače simuliranim kaljenjem, u svakom pokretanju početna temperatura je bila postavljena na 1000. Za svaki par parametara $\alpha, c_{\max}$ napravljeno je 10 pokretanja programa. Rezultate prikazuju tablice 4.1 i 4.2.

Testiranja pokazuju da rezultat nije nužno bolji što je broj iteracija veći. Iz rezultata je vidljivo kako se s većim c_{max} povećava prosjek nađenih rješenja, međutim puni se efekt postiže ako se zajedno s povećanjem broja iteracija poveća koeficijent hlađenja (hlađenje je tada sporije). Obzirom na jednostavnost heuristike, relativno veliki broj potrebnih iteracija može se smatrati očekivanim.

Kod tabu pretraživanja također je za svaku kombinaciju ulaznih parametara program pokrenut 10 puta za dani problem. Inače, kod zahtjevnijih problema, ponekad je potrebno i više pokretanja obzirom na prirodu heuristike. Naime, pretraga prostora rješenja je donekle limitirana inicijalnim rješenjem pa se pokazalo dobrim algoritam pokretati s više različitih početnih rješenja.

Sva testiranja pokazuju da se rezultat poboljšava do oko 200-te iteracije. S većim brojem iteracija u svim je testovima dobiven istovjetan rezultat. Ovo se zapažanje slaže sa zaključcima drugih autora (Kreher, Stinson, 1999, "*...We found c_{max}=200 to be sufficient for the problem instances we considered.*").

Za dani primjer naprtnjače potreno je optimirati duljinu tabu liste. Tablice 4.3 i 4.4 prikazuju rezultate za dva primjera.

L	Min	Max	Prosjek	Broj nađenih optimuma u 10 pokretanja
1	991	1046	1008.9	0
2	991	1040	1008.9	0
3	999	1053	1044.4	3
4	999	1053	1044.2	2
5	1024	1051	1041.5	0
6	1024	1051	1045.1	0
7	1019	1040	1030.4	0
8	1020	1040	1033.2	0

Tablica 4.3 *Rezultati rješavanja problema naptranjače sa* 15 *grumena tabu pretraživanjem, sa* c_{max}=200

L	Min	Max	Prosjek	Broj nađenih optimuma u 10 pokretanja
1	7464	7464	7464	10
2	7464	7464	7464	10
3	7407	7464	7433.3	3
4	7407	7464	7432.5	3
5	7300	7464	7422.9	5
6	7294	7464	7423.2	6
7	7300	7464	7412.2	5
8	7304	7464	7391	2

Tablica 4.4 *Rezultati rješavanja problema naptranjače sa* 25 *grumena tabu pretraživanjem, sa* c_{max}=200

Genetski algoritam je primijenjen s populacijom od 50 vektora i postotkom jedinki određenih za eliminaciju *M* od 50%. Parametri koji su varirali su bili broj iteracija i vjerojatnost mutacije. Vjerojatnost mutacije se kretala od 0.003 do 0.05.

Dulje vrijeme evolucije je donosilo i bolji rezultat, no algoritam je bio osjetljiviji (sukladno općoj karakteristici genetskog algoritma) na vjerojatnost mutacije. Iako je za svaki primjer potrebno optimirati vjerojatnost mutacije, algoritam se pokazao vrlo pouzdanim i od uspoređivanih algoritama najmanje osjetljiv na ulazne parametre. Rezultati su prikazani u tablicama 4.5 i 4.6.

p_m	GA_1 (križanjem s jednom točkom prekida)				GA_2 (uniformno križanje)
	Min	Max	Prosjek	Broj nađenih optimuma u 10 pokretanja	Broj nađenih optimuma u 10 pokretanja
0.05	1040	1053	1050.3	5	3
0.04	1051	1053	1052.6	8	6
0.03	1009	1053	1048.2	7	6
0.01	985	1053	1038.9	2	3
0.007	985	1053	1029.0	1	3
0.005	995	1053	1025.0	1	2
0.003	997	1053	1029.1	2	0

Tablica 4.5 *Rezultati rješavanja problema naptranjače sa* 15 *grumena jednostavnim genetskim algoritmom s eliminacijskom selekcijom, za t*=300*, VEL_POP*=50

p_m	GA_1 (križanjem s jednom točkom prekida)				GA_2 (uniformno križanje)
	Min	Max	Prosjek	Broj nađenih optimuma u 10 pokretanja	Broj nađenih optimuma u 10 pokretanja
0.05	7080	7464	7341.4	2	2
0.04	7249	7464	7389.0	4	0
0.03	7304	7464	7434.1	7	4
0.01	7307	7464	7418.1	5	3
0.007	7137	7411	7292.2	0	3
0.005	6706	7420	7209.8	0	4
0.003	6607	7420	7126.9	0	0

Tablica 4.6 *Rezultati rješavanja problema naptranjače sa* 25 *grumena jednostavnim genetskim algoritmom s eliminacijskom selekcijom, za t*=300*, VEL_POP*=50

Kod primjera dimenzija 15 i 25 uspoređivani genetski algoritmi su podjednako uspješni, dok je kod primjera dimenzija 50 i 100 genetski algoritam s uniformnim križanjem generirao više opitmalnih rješenja u 10 pokretanja.

Za dimenzije 15 i 25 rješenja su provjerena deterministički (prikazuju ih tablice 4.5, 4.6 i prilog B), za dimenziju 50 maksimalna vrijednost funkcije cilja ("najveća vrijednost zlata koje može stati u naprtnjaču") iznosi 15522, a za dimenziju 100 maksimalna vrijednost iznosi 26271. Pritom je rješenje za *n*=50 vrlo vjerojatno optimalno rješenje jer su svi algoritmi došli do tog rješenja a osim toga rješenje je bilo vrlo učestalo kod robusnijeg pokretanja algoritma (primjerice, kod genetskog

algoritma sa uniformnim križanjem pokrenutog s parametrima t=1000, VEL_POP=100, p_m=0.01, 9 od 10 rezultata iznosilo je 15522).

4.4 Zaključak

Kobminatorički problem poznat pod nazivom problem naptrnjače, koji spada u klasu NP teških problema i koji ima značajnu primjenu kod matematičkog modeliranja, rješavan je pomoću heurističkih algoritama s ciljem njihove usporedbe.

Simulirano kaljenje zahtijeva relativno velik broj iteracija a koeficijent hlađenja je potrebno optimirati za dani primjer. Primijećeno je da, kod rješavanja problema s velikim dimenzijama, veći broj iteracija treba pratiti sporije hlađenje (primjerice, rezultat 15522 kod trećeg primjera je dosegnut sa brojem iteracija od 2 milijuna i koeficijentom hlađenja 0.999998; povećanje samo jednog od tih parametara ne dovodi do boljeg rezultata).
Kod tabu pretraživanja nije potreban broj iteracija veći od 200 a duljinu liste je potrebno optimirati za dani primjer. Kako prikazuje tablica 4.7 ovaj algoritam se pokazao najefikasnijim.
Genetski algoritam se pokazao kao najrobusniji te najpouzdaniji za dimenzije do 50. Zabilježen je ponajveći broj nađenih optimalnih rješenja a ostala su bila s malim odstupanjem od optimuma. Algoritam je najosjetljiviji na vjerojatnost mutacije, koju je potrebno optimirati za dani primjer. Primijećeno je da je kod većih dimenzija efikasniji genetski algoritam s uniformnim križanjem, dok su kod manjih dimenzija genetski algoritmi podjednako efikasni.

n	Prostor rješenja (2^n)	Broj izračuna funkcije cilja			
		SA	TS	GA, jednostavni s elim. sel.	GA, s uniform. križanjem
		5000	3000	15000	15000
25	33554432	20000	5000	15000	15000
50	$1.1 \cdot 10^{15}$	2000000	10000	50000	50000
100	$1.2 \cdot 10^{30}$	>100000000	18000	>5000000	>5000000

Tablica 4.7 *Usporedba efikasnosti algoritama na problemu naprtnjače*

Budući da je kod algoritama temeljna operacija, o kojoj najviše ovisi vrijeme rada algoritma, izračun funkcije cilja[9] algoritme smo usporedili po broju izračuna funkcije cilja (jedino je genetski algoritam vremenski još zahtjevniji zbog operacija selekcije, križanja i mutacije).

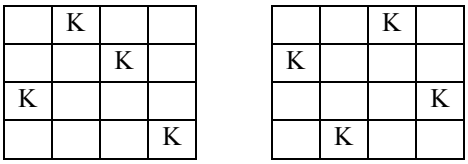
[9] Kod simuliranog kaljenja, broj izračuna funkcije cilja jednak je broju iteracija, kod tabu pretraživanja u svakoj se iteraciji izračuna n-L vrijednosti funkcija profita a kod genetskog algoritma u svakoj se generaciji-iteraciji svakoj jedinki u populaciji računa profit.

5. Usporedba heurističkih algoritama na problemu n-kraljica

U radu se uspoređuju heuristički algoritmi simulirano kaljenje, tabu pretraživanje i genetski algoritam na primjeru problema n-kraljica. Opisane su heuristike i prikazani rezultati testiranja za pojedini algoritam. Izvedeni su zaključci o primjeni heurističkih algoritama na ovoj NP klasi problema, kao i njihova usporedba.

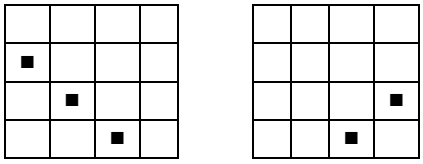
5.1 Problem n-kraljica

Problem 8 kraljica je klasični kombinatorički problem u kojem se traži takav raspored kraljica na šahovskoj ploči da se kraljice međusobno ne napadaju. Ovaj se problem može poopćiti zadavanjem bilo kojeg broja kraljica, odnosno dimenzija šahovske ploče (na šahovski ploču $n \times n$ treba smjestiti n kraljica; n>3), pa govorimo o problemu n-kraljica.
Dok je za manji broj kraljica problem još moguće riješiti deterministčkim metodama, za veći n to postaje nepraktično. Budući da je složenost ovog problema $O(n!)$ koristiti će se heuristički algoritmi.
U svakom od uspoređivanih algoritama, rješenje će se zapisivati u obliku uređene n-torke $(q_1, q_2, ..., q_n)$ koja može biti bilo koja permutacija skupa $(1,2,...,n)$. Pri tome indeks, tj. pozicija broja u vektoru, predstavlja stupac a njegova vrijednost redak u kojem se kraljica nalazi (brojeći odozdo prema gore). Slika 5.1 prikazuje notacije rješenja za problem 4-kraljice.



Slika 5.1 *Primjeri notacije rješenja za problem* 4*-kraljice*

Budući da se prema ovako zapisanom raporedu kraljica one ne mogu napadati horizontalno niti vertikalno, funkcija cilja jedino treba obraćati pozornost na konflikte po dijagonalama. Stoga se funkcija cilja[10] definira na način: ako je na nekoj dijagonali n kraljica, broj konflikata je $n-1$. Prebrojiti treba lijeve i desne dijagonale (sl. 5.2), a funkcija vraća ukupni broj konfikata.



Slika 5.2 *Funkcija cilja vraća broj konflikata po dijagonalama*

[10] Pseudokod funkcije preuzet iz rada Božiković, Golub, Budin, "Solving n-Queen problem using global parallel genetic algorithm", Eurocon 2003, Ljubljana, Slovenija

5.2 Opis heuristika

Kao i u prethodna dva istraživanja i u ovome radu se nastojalo da heuristike budu što je moguće jednostavnije i što sličnije kako bi algoritmi bili što usporedljiviji; dakako vodeći brigu o tome da što uspješnije rješavaju zadani problem.

Element koji se javlja u sva tri algoritma je zamjena dvaju slučajno odabranih pozicija. Točnije, u simuliranom kaljenju taj mehanizam čini cijelu heuristiku, dok se kod tabu pretraživanja pozicije zamjenjuju slijedno kako bi se susjedstvo danog rješenja iscrpno pretražilo. Dakle, kod tabu pretraživanja susjedstvo čine ona rješenja koja se od danog mogu generirati zamjenom svih parova pozicija. Takav odabir čini ovu heuristiku prilično složenom, gornja granica složenosti iznosi $O\left(\frac{(n-1)n}{2}\right)$. Broj izračuna funkcije cilja u pojedinoj iteraciji onoliko je manji je od ovog broja koliko je parova u tabu listi.

Kod genetskog algoritma zamjena dvaju slučajno odabranih pozicija prisutna je u operatoru mutacije. Koristi se 3-turnirska selekcija a križanje se izvodi tako da se podudarne pozicije kod roditelja prenose na dijete a preostale pozicije se generiraju slučajno - pazeći da kromosom bude pravilan[11].

5.3 Rezultati testiranja

Rezultate rješavanja problema n-kraljica simuliranim kaljenjem prikazuje tablica 5.1. Vidljiva je korelacija između broja potrebnih iteracija i broja kraljica. Što je *n* veći to je više iteracija potrebno da bi se generiralo rješenje, no rast je puno sporiji sa *n* nego prostor rješenja (*n!*).
U načelu, paramerar α potrebno je optimirati za pojedini slučaj, međutim pokazalo se da je za sve *n* optimum oko 0.99.

α	c_{max}	n	Broj rješenja u 10 pokretanja
0.99	1000	8	10
		10	10
	5000	30	10
		50	9
	10000	75	10
		100	9
	20000	200	9
		300	3
	30000	400	3
	60000	500	9
		750	1
	100000	1000	5

Tablica 5.1 *Rezultati rješavanja* n-*kraljica simuliranim kaljenjem,* T_0=1000

[11] Prema Božiković, Golub, Budin, "Solving n-Queen problem using global parallel genetic algorithm", Eurocon 2003, Ljubljana, Slovenija

Kod tabu pretraživanja primjetno je da je bolje kad su za veći n veći i L i broj iteracija. Primjerice, za n>=300 nije bilo moguće pronaći traženi raspored kraljica sa c_{max}=100. Rezultate prikazuje tablica 5.2.

L	c_{max}	n	Broj rješenja u 10 pokretanja
3	100	8	6
		10	5
		30	7
		50	5
		75	6
		100	6
		200	2
5	200	300	2
8	300	400	2 (od 4 pokretanja)

Tablica 5.2 *Rezultati rješavanja* n-*kraljica tabu pretraživanjem*

Opisani genetski algoritam pokrenut je s populacijom od 100 jedinki i vjerojatnošću mutacije 0.02, dok je kao uvjet završetka evolucije postavljeno nalaženje traženog rješenja (tj. da funkcija cilja vrati nulu). Rezultate prikazuje tablica 5.3.

n	Broj iteracija do prvog rješenja
8	3
10	63
30	814
50	4026
75	2669
100	11543
200	36093
300	117356

Tablica 5.3 *Rezultati rješavanja n-kraljica genetskim algoritmom, za* p_m=0.02*, VEL_POP*=100

5.4 Zaključak

I kod ovog je problema temeljna operacija, o kojoj najviše ovisi vrijeme rada algoritma, izračun funkcije cilja. Kod simuliranog kaljenja broj izračuna funkcije cilja jednak je broju iteracija, dok su kod ostala dva algoritma heuristike složenije te u pojedinom koraku funkciju cilja računaju više puta.
Simulirano kaljenje je jedini algoritam koji je u praktičnom vremenu sposoban riješiti probleme s vrlo velikim brojem kraljica. Isto je moguće postići i genetskim algoritmom ako se koriste paralelna računala (Božiković, Golub, Budin, 2003).

Algoritme smo usporedili po broju izračuna funkcije cilja (tablica 5.4). Pokazalo se da je simulirano kaljenje najefikasniji algoritam. Kod tog algoritma broj izračuna funkcije cilja najsporije raste s dimenzijom problema.

n	Prostor rješenja ($n!$)	Broj izračuna funkcije cilja		
		SA	TS	GA
8	40320	1000	2800	300
10	3628800	1000	4500	6300
30	$2.6 \cdot 10^{32}$	5000	43500	81400
50	$3.0 \cdot 10^{64}$	5000	122500	402600
75	$2.4 \cdot 10^{109}$	10000	277500	266900
100	$9.3 \cdot 10^{157}$	10000	495000	1154300
200	...	20000	1990000	3609300
300		20000	8970000	11735600
400		30000	23940000	
500		60000		
750		60000		
1000		100000		

Tablica 5.4 *Usporedba algoritama na problemu n-kraljica*

6. Zaključak

Rad se bavi usporedbom heurističkih agloritama na tri klase problema. Heuristički algoritmi penjanje uzbrdo, simulirano kaljenje, tabu pretraživanje i genetski algoritam uspoređeni su na primjeru optimiranja realne funkcije jedne varijable. Na NP teškim problemima naprtnjače i n-kraljica uspoređeni su algoritmi simulirano kaljenje, tabu pretraživanje i genetski algoritmi.

Za svaku od tri odabrane klase problema najprije su osmišljene i implementirane heuristike, nakon čega su provedeni usporedni testovi te izvedeni zaključci. Pri kreiranju heuristika pokazalo se da konceptualno vrlo jednostavne ideje mogu biti vrlo uspješne u rješavanju danog problema (kao, primjerice, kad susjedstvo čine vektori s Hammingovom udaljenošću jednakom 1). Također je za primijetiti da iste heuristike mogu biti primjenjive na različitim problemima. Tako kod simuliranog kaljenja, heuristika koja od danog rješenja novog kandidata generira promjenom slučajno odabranog bita, uspješno rješava i optimiranje funkcija i problem naprtnjače. Analogno vrijedi i za genetski algoritam, kojem je to opća karakteristika.

Kod optimiranja funkcija sva četiri uspoređivana algoritma pokazala su se vrlo uspješna, došavši do rješenja tražene točnosti u većini pokretanja. Budući da je od tri klase problema optimiranje funkcija bilo najmanje zahtjevno, ovdje su algoritmi uspoređivani po kriterijima efikasnosti (broj potrebnih iteracija/izračuna vrijednosti funkcije cilja) i pouzdanosti (vjerojatnost da je dobiveno rješenje optimalno, te očekivano odstupanje najlošijeg rješenja). Usporedne rezultate prikazuje tablica 3.12. Kao najefikasniji, najprecizniji i najpouzdaniji pokazao se algoritam penjanje uzbrdo, koji je ujedno i konceptualno najjednostavniji među analiziranim algoritmima. Za sve testirane funkcije, penjanjem uzbrdo, optimumi su nađeni u svakom pokretanju na točnost od 15 decimala u svega nekoliko iteracija.
Primijećeno je, sukladno očekivanju, da je globalni optimum to teže pronaći što je omjer širine prostora rješenja na kojem se nalazi globalni optimum i prostora rješenja manji i što je vrijednost lokalnih optimuma bliža vrijednosti globalnog optimuma.

Kod problema naprtnjače i problema n-kraljica algoritmi su uspoređivani prema broju izračuna funkcije cilja za zadanu dimenziju problema, budući da o toj operaciji najviše ovisi vrijeme rada algoritma. Izuzetak po tom pitanju je jedino genetski algoritam koji još dodatno zahtijeva vrijeme za križanje i druge operacije. Za problem naprtnjače sumarne rezultate prikazuje tablica 4.7.

Kod problema n-kraljica najefikasnijim se pokazalo simulirano kaljenje. Jedino je tim algoritmom bilo moguće riješiti probleme s velikim brojem kraljica (1000) u praktičnom vremenu. Usporedne testove, koji prikazuju ovisnost broja izračuna funkcije cilja o dimenziji problema, za pojedini algoritam, prikazuje tablica 5.4.

Kod genetskih algoritama, uniformno križanje se pokazalo uspješnije od križanja s jednom točkom prekida. Svi testovi potvrđuju opće karakteristike genetskih algoritama kao što su univerzalnost u rješavanju problema, pouzdanost i osjetljivost na vjerojatnost mutacije.

Svi testirani algoritmi pokazali su se uspješnima u rješavanju sve tri klase problema. Na optimiranju funkcija uspređivane su njihove pouzdanosti, preciznosti i efikasnosti te su optimirani ulazni parametri. Na problemu naprtnjače i problemu n-kraljica, koji predstavljaju NP teške probleme uspređene su efikasnosti algoritama te ispitani njihovi dosezi za velike dimenzije problema. Svi su testirani heuristički algoritmi pokazali gornju granicu složenosti znatno manju do deterministčkih metoda za ove probleme.

Literatura

[1] Kreher, D.L., Stinson, D.R., *Combinatorial algorithms*, CRC Press, New York, 1999.

[2] Golub, Marin, *Genetski algoritam, prvi dio*, http://www.zemris.fer.hr/~golub/ga/ga.html (22.05.2006.), Faculty of Electrical Engineering and Computing, 2004.

[3] Golub, Marin, *Genetski algoritam, drugi dio*, http://www.zemris.fer.hr/~golub/ga/ga.html (22.05.2006.), Faculty of Electrical Engineering and Computing, 2004.

[2] Ivanšić, Ivan, *Numerička matematika*, Element, Zagreb, 1998.

[3] Pavković, B., Dakić, B., *Polinomi*, Školska knjiga, Zagreb, 1990.

[4] Kurepa, S., *Matematička analiza, funkcije jedne varijable*, Tehnička knjiga, Zagreb, 1990.

[5] Golub, Marin, *Vrednovanje uporabe genetskih algoritama za aproksimaciju vremenskih nizova*, magistarski rad, FER, Zagreb, 1996.

[6] Božiković, Marko, *Globalni paralelni genetski algoritam*, diplomski rad, FER, Zagreb, 2000.

[7] Božiković, M., Golub, M., Budin, L., *Solving n-Queen problem using global parallel genetic algorithm*, konferencija Eurocon, Ljubljana, Slovenija, 2003.

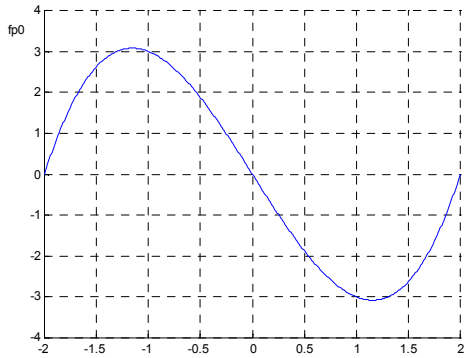
Prilog A: Ispitne funkcije

a) popis ispitnih funkcija

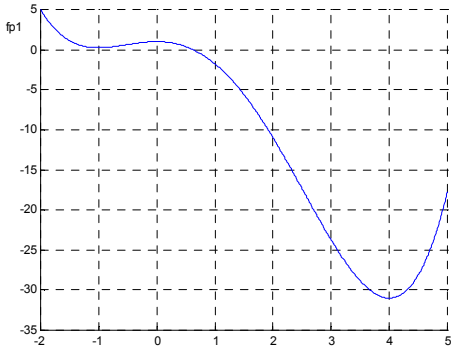
Funkcija	Interval	f_{opt} 6 decimala	f_{opt} 15 decimala
$f_{p0}(x) = x^3 - 4x$	[-2, 2]	3.079201	3.079201435678004
$f_{p1}(x) = \frac{1}{4}x^4 - x^3 - 2x^2 + 1$	[-2, 5]	-31	-31
$f_{p2}(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$	[-1, 1]	-0.428571	0.428571428571429
$f_0(x) = 1 + x\sin(10\pi x)$	[-1, 2]	2.850275	2.850275329628810
$f_1(x) = \frac{\sin(10\pi x^2)}{x}$	[-1, 2]	4.771197	4.771196687322133
$f_2(x) = x^2 \sin(10\pi x^2)$	[1, 2]	3.850135	3.850134829778519
$f_3(x) = x^{\sin(x)}$	[0, 50]	45.555967	45.555967313423452
$f_4(x) = x\cos[tg(x)]$	[-2, 6]	5.747734	5.747734006936660
$f_5(x) = x\cos(tg\sqrt{x})$	[0, 5]	2.989384	2.989384266778107
$f_6(x) = x\sin\left\{\frac{30}{x}\left[\frac{\pi}{2} - arctg(x)\right]\right\}$	[-1, 8]	4.632533	4.632532808974171
$f_7(x) = 0.5 - \frac{\sin^2(x) - 0.5}{(1 + 0.001x^2)^2}$	[-100, 100]	1	1

b) grafovi ispitnih funkcija

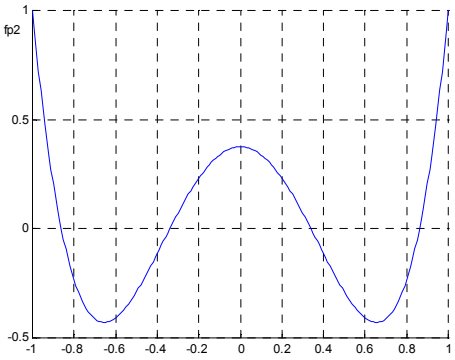
$$f_{p0}(x) = x^3 - 4x$$



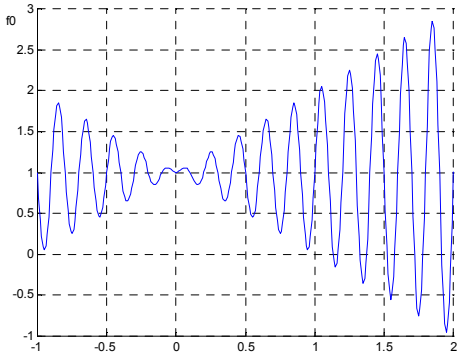
$$f_{p1}(x) = \frac{1}{4}x^4 - x^3 - 2x^2 + 1$$



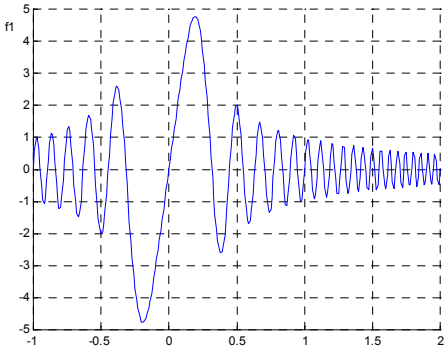
$$f_{p2}(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$$



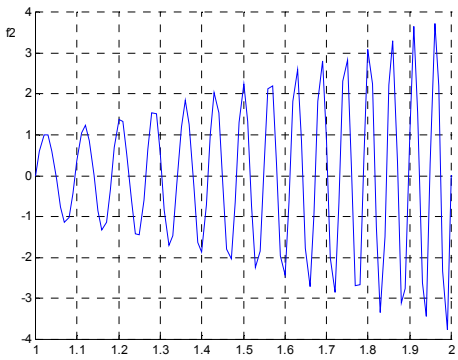
$$f_0(x) = 1 + x\sin(10\pi x)$$



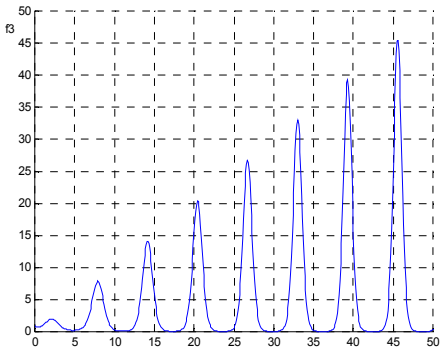
$$f_1(x) = \frac{\sin(10\pi x^2)}{x}$$



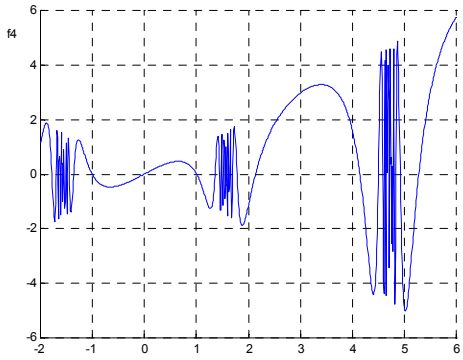
$$f_2(x) = x^2 \sin(10\pi x^2)$$



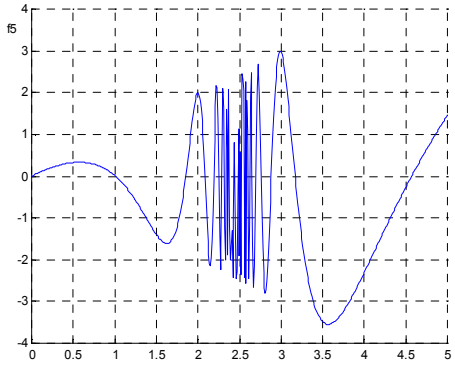
$$f_3(x) = x^{\sin(x)}$$



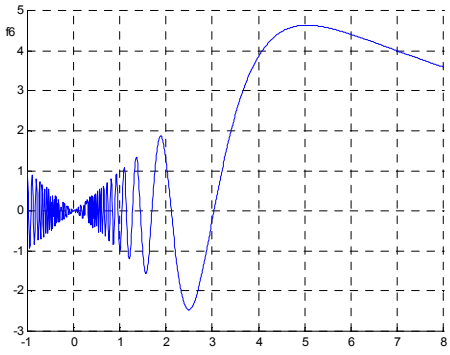
$$f_4(x) = x\cos[tg(x)]$$



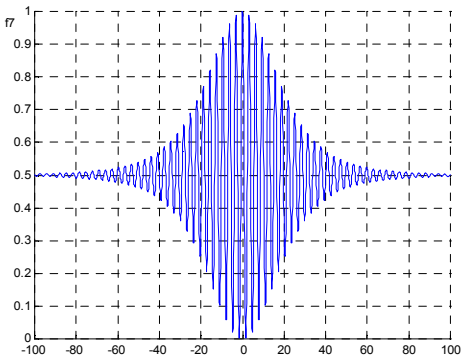
$$f_5(x) = x\cos(tg\sqrt{x})$$



$$f_6(x) = x\sin\left\{\frac{30}{x}\left[\frac{\pi}{2} - arctg(x)\right]\right\}$$



$$f_7(x) = 0.5 - \frac{\sin^2(x) - 0.5}{(1 + 0.001x^2)^2}$$



Prilog B: Ispitni primjeri problema naprtnjače

Profiti	51 54 57 60 62 68 70 75 78 96 101 103 105 108 118
Težine	117 122 128 131 133 141 152 167 172 181 192 194 203 229 220
Kapacitet	480
Optimalno rješenje	X=[1 1 1 0 1 0 1 0 1 0 0 0 0 1 0]
Maksimalni profit	1053

Primjer problema naprtnjače sa 15 grumena

Profiti	845 456 45 987 345 983 1503 798 987 1023 1200 1230 194 4 1232 987 1244 789 1245 1455 1467 1500 1501 723 1530
Težine	545 456 145 197 345 483 103 698 487 523 600 430 394 567 232 887 344 789 545 455 467 500 501 323 530
Kapacitet	10340
Optimalno rješenje	X=[1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0]
Maksimalni profit	7464

Primjer problema naprtnjače sa 25 grumena

Profiti	845 456 45 987 345 983 1503 798 987 1023 1200 1230 194 4 1232 987 1244 789 1245 1455 1467 1500 1501 723 1530 844 457 45 937 395 913 1403 778 937 1023 1250 1230 174 14 1242 927 294 789 1245 1455 1467 1500 1561 723 1430
Težine	545 456 145 197 345 483 103 698 487 523 600 430 394 567 232 887 344 789 545 455 467 500 501 323 530 545 456 145 197 395 483 103 698 487 563 600 430 394 567 232 887 394 789 655 455 427 530 551 323 570
Kapacitet	20130
Optimalno rješenje	[11101101111011010110000001110110111101101111000000]
Maksimalni profit	15522

Primjer problema naprtnjače sa 50 grumena

Profiti	192 446 278 315 166 380 127 299 56 332 102 91 10 345 387 96 36 171 242 456 192 430 443 253 196 399 174 435 399 445 326 187 457 334 435 383 32 341 499 460 93 112 181 87 90 206 306 322 88 311 163 454 224 424 151 327 259 239 4 300 222 187 245 228 339 51 439 176 432 371 280 125 107 56 479 348 475 19 19 255 436 253 266 81 107 429 328 197 59 150 135 304 117 97 34 81 31 483 182 409
Težine	372 911 588 675 305 787 229 568 102 652 190 189 21 637 846 205 77 368 522 983 380 804 941 459 407 802 352 889 759 951 642 342 984 727 950 768 68 723 1021 865 169 227 373 161 167 434 647 643 170 633 351 845 429 896 272 600 509 508 8 645 488 392 470 497 691 91 941 353 873 809 557 256 199 103 869 646 921 40 40 501 933 481 514 148 196 871 680 384 127 306 296 574 218 201 67 147 57 1060 336 898
Kapacitet	12360
Optimalno rješenje	[01110100000110111111001010010100111011100110011000100100011111011010010100000110100000101010010001 01]
Maksimalni profit	26271

Primjer problema naprtnjače sa 100 grumena

Prilog C: Rješenje 500-kraljica

309 243 255 445 3 469 218 284 457 464 129 357 412 405 330 220 84 36 242 103 178
168 259 333 219 241 373 79 250 88 415 254 223 306 383 325 292 482 107 138 136
70 181 249 385 102 324 169 98 268 82 411 44 355 229 171 86 321 317 364 135 313
307 26 361 481 20 227 247 334 164 261 85 11 187 369 374 406 471 393 475 377 452
146 234 282 286 271 50 199 344 375 299 104 74 303 132 225 53 142 339 350 395 55
121 288 488 435 93 191 214 403 239 57 176 216 280 289 463 189 366 116 342 230
89 61 465 408 461 30 112 367 336 212 474 296 473 130 161 123 443 49 253 5 59
311 125 448 391 120 400 97 478 101 235 113 29 37 165 394 48 66 87 15 293 65 301
109 222 141 210 2 345 459 491 16 354 421 444 240 231 92 17 122 467 356 266 19
290 456 388 195 119 431 58 152 470 137 347 368 500 35 285 154 134 462 25 246
332 224 499 90 429 331 359 149 238 291 274 207 414 32 99 438 494 446 489 13 304
417 162 387 118 251 279 153 143 399 108 208 260 63 453 315 38 365 480 114 22 46
64 363 155 83 42 205 305 439 75 226 404 275 442 67 322 139 183 460 449 401 495
56 170 43 197 287 427 117 124 346 248 12 484 386 349 295 422 402 441 202 281 68
51 450 188 209 252 278 396 340 159 54 479 420 409 47 193 433 18 33 458 312 497
300 6 407 211 283 320 308 472 8 455 77 351 206 184 323 167 419 140 454 425 7 39
41 466 451 338 71 486 80 151 430 390 21 95 1 203 397 327 217 437 180 126 148 9
392 389 244 23 358 297 72 215 131 410 96 233 447 14 158 493 270 294 10 245 182
27 228 341 94 424 314 492 52 376 185 204 436 423 172 263 62 370 175 329 257 78
496 111 196 105 179 156 326 477 380 434 145 384 166 316 468 110 360 45 60 100
483 310 258 381 426 343 81 76 265 174 262 198 157 4 133 150 372 69 273 432 160
213 40 318 413 267 498 362 236 31 115 256 186 416 352 34 192 201 353 147 371
485 28 490 221 277 319 487 144 476 173 237 328 272 379 337 177 378 302 200 106
276 428 190 91 73 24 269 335 128 398 127 440 232 382 418 264 194 298 348 163

Jedno od rješenja 500-kraljica generirano simuliranim kaljenjem

Prilog D: Praktični rad (CD)

Praktični rad sastoji se od implementacija opisanih algoritama u programskom jeziku C++ te od grafova ispitnih funkcija koji su izrađeni pomoću računalnog programa Matlab. Sve kreirane datoteke nalaze se na pratećem CD-u.

Popis praktičnih radova (sadržaj CD-a):

- Optimiranje funkcija
  - Konzolne aplikacije
  - Izvorni kod
  - Datoteke "ReadMe" s tehničkim uputama
- Problem naprtnjače
  - Konzolne aplikacije
  - Izvorni kod
  - Datoteke "ReadMe" s tehničkim uputama
- Problem naprtnjače
  - Konzolne aplikacije
  - Izvorni kod
  - Datoteke "ReadMe" s tehničkim uputama
- Grafovi ispitnih funkcija
  - Grafovi 11 testiranih funkcija u formatima .emf i .fig
- Funkcije cilja
  - Izdvojeni izvorni kod funkcija cilja
- Rješenja n-kraljica
  - Datoteke s rješenjima od 8 do 1000 kraljica