UNIVERZITET U NIŠU
PRIRODNO - MATEMATIČKI FAKULTET
DEPARTMAN ZA RAČUNARSKE NAUKE

IVONA I. BRAJEVIĆ

POBOLJŠANJA NEKIH POPULACIONIH METAHEURISTIKA ZA REŠAVANJE OPTIMIZACIONIH PROBLEMA SA OGRANIČENJIMA

DOKTORSKA DISERTACIJA

NIŠ, 2015

UNIVERSITY OF NIŠ
FACULTY OF SCIENCES AND MATHEMATICS
DEPARTMENT OF COMPUTER SCIENCE

IVONA I. BRAJEVIĆ

IMPROVEMENTS OF SOME POPULATION-BASED METAHEURISTICS FOR CONSTRAINED OPTIMIZATION PROBLEMS

PHD THESIS

NIŠ, 2015

KOMISIJA ZA ODBRANU TEZE

MENTOR: **Dr Jelena Ignjatović**
(ČLAN KOMISIJE) Vanredni profesor
Prirodno-matematički fakultet
Univerzitet u Nišu

PREDSEDNIK KOMISIJE: **Dr Miroslav Ćirić**
Redovni profesor
Prirodno-matematički fakultet
Univerzitet u Nišu

ČLAN KOMISIJE: **Akademik dr Gradimir Milovanović**
Redovni član SANU
Srpska akademija nauka i umetnosti
Beograd

ČLAN KOMISIJE: **Dr Predrag Stanimirović**
Redovni profesor
Prirodno-matematički fakultet
Univerzitet u Nišu

DATUM ODBRANE: _____

Uvod

Mnogi optimizacioni problemi sa ograničenjima predstavljaju teške probleme. Njihovo rešavanje klasičnim algoritmima daje rešenja po eksponencijalno višoj ceni vremena izračunavanja, kako dimenzije problema rastu. Nekada je moguće dobiti tačno rešenje, ali se često, zavisno od prirode problema, dobijaju samo približna rešenja. Razvoj računara nije doprineo da se ovo promeni, jer mnogi optimizacioni problemi ostaju nedovoljno dobro rešivi klasičnim algoritmima. Međutim, korišćenje računara za realizaciju simulacije određenih procesa u prirodi, pružilo je veću mogućnost da se rešavanju teških optimizacionih problema pristupi i na drugi način.

Praćenjem osnovnih koraka nekih prirodnih fenomena nastaju algoritmi inspirisani prirodom. Njihovom realizacijom kroz računarske programe i proverom dobijenih rezultata pokazalo se da uspešno rešavaju mnoge teške optimizacione probleme za relativno kratko vreme. Razvoj algoritama za rešavanje optimizacionih problema se na taj način kreće u dva smera.

Prvi, deterministički, koji će pri svakom izvršavanju u bilo kojim uslovima od istih ulaznih promenljivih doći do istih izlaznih rezultata prateći svaki put identičan niz naredbi. Ovi algoritmi podrazumevaju značajna numerička izračunavanja i na kvalitet dobijenih rezultata utiče činjenica da oni zahtevaju prilagođavanje problema njihovim specifičnim procedurama. Drugu grupu algoritama čine heurističke metode, koje bar u jednom delu izvršavanja donose neku odluku slučajnom komponentom. Ne postoji garancija da će rezultati dobijeni primenom ovih tehnika biti optimalni i dopustivi, ali je njihova prednost da rešavaju optimizacione probleme za relativno kratko vreme izračunavanja, pri čemu se mogu prilagođavati razmatranim problemima.

Sa namerom da se heuristički algoritmi, koji su usko prilagođeni rešavanju pojedinačnih problema, učine opštim, sa mogućnostima da globalno reše širi skup problema, razvijaju se metaheuristike. Metaheuristike, nove ili moderne heuristike, su prirodom inspirisani algoritmi koji se poslednjih decenija ubrzano razvijaju, sa tendencijom da se taj razvoj nastavi. To je zahtevan

proces kada se uzme u obzir da metaheuristike sadrže različite komponente čiji uticaj treba istražiti.

U ovoj disertaciji izložene su poboljšane varijante nekih istaknutih populacionih metaheuristika za rešavanje optimizacionih problema sa ograničenjima. Posebno su razmatrane: metaheuristika bazirana na pčelinjim kolonijama, algoritam svitaca, pretraga kukavice i metaheuristika bazirana na populaciji ljudske grupe. Uočeno je da se poboljšanja mogu postići modifikacijom jednačina pretrage ili kombinovanjem jednačina pretrage različitih metaheuristika, korišćenjem novih operatora selekcije, kao i finim podešavanjima parametara koji kontrolišu izvršavanje algoritma. Predložene poboljšane varijante populacionih algoritama su testirane na velikom skupu standardnih benčmark problema i za rešavanje nekih realnih problema koji predstavljaju optimizacione probleme sa ograničenjima.

Prvi deo disertacije obuhvata osnovne pojmove vezane za optimizacione probleme sa ograničenjima, metode rešavanja optimizacionih problema i detaljniji uvid o istaknutim metaheuristikama za koje se uvode poboljšanja.

U drugom delu disertacije je predstavljena poboljšana varijanta metaheuristike bazirane na kolonijama pčela. Predloženi algoritam uvodi pet modifikacija u odnosu na originalni algoritam čime značajno menja funkcionisanje originalnog algoritma. Osnovne modifikacije se odnose na korišćenje dva modifikovana operatora pretrage u istraživačkoj i posmatračkoj fazi i korišćenje uniformnog operatora ukrštanja u izviđackoj fazi. Predloženi algoritam je testiran na dvadeset osam standardnih benčmark funkcija i upoređen sa četrnaest odgovarajućih metaheuristika.

U trećem delu disertacije je izložen hibridni algoritam pretraživanja ljudske grupe za rešavanje problema globalne optimizacije. U cilju poboljšanja performansi metaheuristike bazirane na populaciji ljudske grupe, kreiran je hibridni algoritam koji uključuje jednačine pretrage metaheuristike bazirane na pčelinjim kolonijama u algoritam baziran na populaciji ljudske grupe. Testiran je na standardnom skupu od dvadeset tri benčmark funkcije i upoređen sa pet odgovarajućih algoritama.

U četvrtom delu disertacije predložena je poboljšana varijanta algoritma svitaca za rešavanje problema strukturne optimizacije sa varijablama kombinovanih tipova. Uvedene su dve modifikacije u odnosu na originalni algoritam svitaca. Jedna se odnosi na korišćenje operatora selekcije koji je baziran na tri pravila dopustivosti, a druga na korišćenje šeme za dinamičku redukciju parametara koji kontrolišu pretragu. Predloženi algoritam je testiran na četiri problema strukturnog dizajna i upoređen sa osnovnim algoritmom.

U petom delu disertacije je prikazana adaptacija pretrage kukavice i algo-

ritma svitaca za rešavanje problema segmentacije slika tehnikom trešholding. Za poređenje rezultata korišćene su metaheuristike diferencijalne evolucije i rojeva čestica. Kao funkcije cilja korišćene su Kapurova entropija i međuklasna varijansa. Ova četiri algoritma su testirana na standardnim benčmark slikama sa poznatim globalnim optimumima dobijenim metodom iscrpne pretrage do pet nivoa segmentacije.

U zaključku je izložen opšti pregled rezultujućih poboljšanja populacionih metaheuristika predstavljenih u radu, kao i generalna analiza komponenti koje na ta poboljšanja utiču.

Želim da se kao autor ovog rada zahvalim svima koji su mi u tome pomogli. Zahvaljujem svom izuzetnom mentoru, prof. dr Jeleni Ignjatović, koja mi je neiscrpnim stručnim kvalitetima omogućila realizaciju disertacije. Zahvaljujem prof. dr Miroslavu Ćiriću i prof. dr Predragu Stanimiroviću, čiji su mi saveti, primedbe i sugestije bili od neprocenjivog značaja. Zahvaljujem akademiku prof. dr Gradimiru Milovanoviću na nesebičnoj podršci i velikom doprinosu razvoju konkretne istraživačke i naučne misli uopšte. Zahvaljujem saradnicima na projektu III44006, Ministarstva za obrazovanje, nauku i tehnološki razvoj Republike Srbije. Zahvaljujem Branislavu i sestri Danijeli, kao i svojim roditeljima, kolegama i prijateljima za svu pažnju i strpljenje koje mi pružaju.

Sadržaj

Glava 1

Osnovni pojmovi

Glava 1 se sastoji od pet odeljaka. U Odeljku 1.1 uvedena je opšta forma optimizacionog problema. Klasifikacija optimizacionih problema prema različitim kriterijumima data je Odeljku 1.2. U Odeljku 1.3 dat je kratak prikaz metoda koje se koriste za rešavanje optimizacionih problema. Ovde su specijalno naglašeni nedostaci determinističkih pristupa i heurističkih algoritama zbog kojih dolazi do razvitka metaheurističkih algoritama. Osnovna svojstva metaheuristika i njihova klasifikacija prema načinu kreiranja rešenja je data u Odeljku 1.4. Osnovna svojstva i pregled nekih populacionih metaheuristika data su u Odeljku 1.5. Ovde su dodatno opisane i najvažnije metode za upravljanje ograničenjima kod populacionih metaheuristika. Poznavanje ovih tehnika je neophodno za dalji rad, jer se one integrišu u osnovne populacione metaheuristike i imaju značajan uticaj na njihove performanse.

1.1. Opšta forma optimizacionog problema sa ograničenjima

Većina optimizacionih problema se može zapisati u sledećoj formi [107]:

Pronaći vektor optimizacionih promenljivih, $x = (x_1, x_2, \ldots, x_D) \in R^D$ koji minimizuje funkciju cilja $f(x)$, pod ograničenjima:

$$\begin{aligned} g_j(x) &\leq 0, \quad j = 1, \ldots, q \\ h_j(x) &= 0, \quad j = q+1, \ldots, m \end{aligned} \tag{1.1}$$

Pri čemu su $g_j(x)$ ograničenja tipa nejednakosti, $h_j(x)$ ograničenja tipa jednakosti, q je broj ograničenja tipa nejednakosti, $m-q$ je broj ograničenja tipa jednakosti. Svaki parametar, x_i, $i = 1, 2, ..., D$, je ograničen svojom

donjom i gornjom granicom $l_i \leq x_i \leq u_i$ koje definisu prostor pretrage $S \subseteq R^D$.

Iako standardna forma optimizacionog problema sa ograničenjima uključuje samo probleme minimizacije, problem maksimizacije može da se konvertuje u problem minimizacije ako pomnožimo funkciju cilja sa -1. Pored toga, standardna forma optimizacionog problema uključuje samo nejednakosti koje su u obliku manje ili jednako ($\leq$). Ako je ograničenje u obliku nejednakosti veće ili jednako ($\geq$), ono se može zapisati u obliku nejednakosti manje ili jednako množenjem obe strane nejednakosti sa -1.

Proizvoljno rešenje $x \in S$ optimizacionog problema sa ograničenjima naziva se dopustivo rešenje (eng. feasible solution) problema ukoliko zadovoljava sva ograničenja. Deo prostora pretrage $F \subseteq S$ koji obuhvata dopustiva rešenja zovemo dopustiv skup (eng. feasible region) rešenja. Dopustiv skup rešenja se bez umanjenja opštosti uvek može definisati preko ograničenja tipa nejednakosti jer se proizvoljno ograničenje oblika jednakosti $h_j(x) = 0$ može zameniti nejednakostima $|h_j(x)| - \epsilon \leq 0$, gde je $\epsilon > 0$ dozvoljena tolerancija (veoma mali broj). Sve tačke izvan dopustivog skupa su nedopustiva rešenja.

Ako u proizvoljnoj tački $x \in F$ važi $g_j(x) = 0$, onda se ograničenje g_j naziva *aktivno* ograničenje u dopustivom rešenju x, a *neaktivno* ako je $g_j(x) < 0$. Ograničenja tipa jednakosti h_j se nazivaju *aktivnim* u svim tačkama $x \in S$.

1.2. Klasikacija optimizacionih problema

Mnoge klase problema mogu da se posmatraju kao podklase problema optimizacije. Klasifikacija optimizacionih problema se može izvršiti prema različitim kriterijumima.

Jedan od kriterijuma za klasifikaciju ovih problema je broj ograničenja. Problem koji ima samo funkciju cilja, ali nema nikakva ograničenja, se naziva optimizacioni problem bez ograničenja (eng. unconstrained optimization problem). U literaturi se često problemi koji imaju samo ograničenja vezana za granice optimizacionih parametara (eng. bound-constrained problem) nazivaju takođe optimizacionim problemima bez ograničenja. U slučaju da u jednačini 1.1 važi $m = q$, problem ima isključivo ograničenja tipa nejednakosti (eng. inequality-constrained problem). Ukoliko u jednačini 1.1 važi $q = 0$ i $m > 0$, problem ima isključivo ograničenja tipa jednakosti (eng. equality-constrained problem).

Optimizacione probleme možemo klasifikovati na osnovu vrste funkcije cilja i vrsta nejednakosti ograničenja. Problemi linearnog programiranja su

optimizacioni problemi u kojima je funkcija cilja linearna i dopustiv skup zadat linearnim jednačinama i nejednačinama. Kod problema kvadratnog programiranja funkcija cilja osim linearnih može imati i kvadratne izraze, dok dopustiv skup mora biti zadat linearnim jednačinama i nejednačinama. Opšti problemi optimizacije sa ograničenjima kod kojih je funkcija cilja nelinearna ili je bar jedno ograničenje zadato nelinearnom nejednačinom, nazivaju se problemima nelinearnog programiranja (eng. nonlinear optimization problems).

Klasifikacija optimizacionih problema se može izvršiti i u zavisnosti od toga da li su mu optimizacione promenljive neprekidne ili diskretne vrednosti. Optimizacioni problem u kome su sve promenljive neprekidne vrednosti nazivamo problem neprekidne optimizacije. Ukoliko su sve promenljive diskretne vrednosti optimizacioni problem nazivamo problem kombinatorne optimizacije. Kod velikog broja problema kombinatorne optimizacije prostor pretrage je konačan skup, pa se ovaj problem u literaturi često definiše kao problem određivanja ekstremne vrednosti funkcije na konačnom dopustivom skupu. Postoje optimizacioni problemi koji imaju i diskretne i neprekidne promenljive.

Optimizacioni problem može da ima više od jedne funkcije cilja za koje treba naći minimum ili maksimum. Prema broju funkcija cilja optimizacione probleme možemo svrstati u dve kategorije: problemi koji imaju jednu funkciju cilja (eng. single objective optimization problems) i problemi koji imaju više od jedne funkcije cilja (eng. multiobjective optimization problems).

1.3. Metode rešavanja optimizacionih problema

Načini rešavanja problema optimizacije se prema prirodi algoritma mogu podeliti u dve kategorije: deterministički algoritmi i stohastički algoritmi [107].

Deterministički algoritmi su algoritmi koji će pri svakom izvršavanju u bilo kojim uslovima od istih vrednosti ulaznih parametara doći do istih vrednosti izlaznih parametara prateći svaki put identičan niz naredbi. Većina klasičnih algoritama spada u kategoriju determinističkih algoritama. Ovi algoritmi su uglavnom kompleksni i zahtevaju znatna numerička izračunavanja.

Deterministički algoritam iscrpne pretrage ili metod grube sile (eng. exaustive search or brute force) je najosnovnija metoda nalaženja optimalnog rešenja optimizacionih problema. Baziran je na ideji da se sistematski proveri

da li svako potencijalno rešenje zadovoljava problem. Iscrpna pretraga sa sigurnošću nalazi optimalno rešenje. Dobra osobina ovih algoritama je laka implementacija. Sa druge strane, složenost izračunavanja ove metode srazmerna je broju kandidata rešenja. U mnogim praktičnim problemima sa povećanjem veličine problema ovaj metod ima tendenicju veoma brzog rasta složenosti izračunavanja. Iz ovih razloga praktična upotrebna vrednost iscrpne pretrage je ograničena na probleme malih dimenzija.

Primene determinističkih metoda za rešavanje teških optimizacionih problema su nedovoljno efikasne zbog nefleksibilnosti njihove adaptacije prilikom traženja rešenja datog problema [7]. Strategije traženja rešenja koje se koriste u ovim tehnikama su zavisne od vrste funkcije cilja, vrste ograničenja, tipa promenljivih, broja promenljivih i broja ograničenja, kao i same strukture prostora pretrage određenog problema. Sa druge strane, prilagođavanje velikog broja optimizacionih problema specifičnim procedurama koje se koriste u deterministčkim optimizacionim tehnikama nekada nije jednostavno izvršiti, što svakako ima uticaj na kvalitet optimizacionih rezultata. Dodatno, veliki broj determinističkih algoritma je efikasan prilikom traženja lokalnog optimuma, stoga prilikom pretrage uvek postoji rizik da će se ovi algoritmi zaglaviti u lokalnom optimumu [18].

Standardan način da se prevaziđu nedostaci determinističkih metoda rešavanja optimizacionih problema je da se u algoritam uvede neka stohastička komponenta, što omogućava algoritmu izlaženje iz lokalnog optimuma. Algoritmi koji bar u jednom delu izvršavanja donose neku odluku slučajnim odabirom nazivaju se stohastički algoritmi. Postoje i hibridni algoritmi, koji predstavljaju kombinaciju determinističkih i stohastičkih algoritama. Imajući u vidu da ovi algoritmi imaju slučajnu komponentu, oni se u literaturi svrstavaju u kategoriju stohastičkih algoritama. Razlikujemo dva tipa stohastičkih algoritama: heuristike i metaheuristike.

Heuristike su algoritmi koji traže dovoljno dobro rešenje problema za relativno kratko vreme. Kratko vreme pretrage je važna karakteristika ovih tehnika imajući u vidu da je za rešavanje teških optimizacionih problema velikog obima uglavnom nemoguće naći optimalno rešenje u razumnom vremenu. Teorijske garancije da će heuristike pronaći optimalno rešenje, često čak i rešenje koje je blisko optimalnom rešenju, ne postoje. Međutim, bez obzira na nedostatak teorijske podrške, značajan uspeh heuristika pri rešavanju različitih optimizacionih problema je potvrđen eksperimentalnim putem.

Osnovni nedostatak heurističkih tehnika je da su one razvijene za rešavanje jednog određenog problema, pri čemu se traganje za kvalitetnim

rešenjem oslanja na karakteristike konkretnog problema [10]. Imajući u vidu da konkretan problem treba detaljno istražiti u cilju upoznavanja njegovih karakteristika kreiranje heurističkog algoritma je dugotrajan posao.

1.4. Metaheuristike

Zbog ograničene primenljivosti determinističkih i heurističkih pristupa, poslednjih decenija dolazi do razvoja velikog broja metaheuristika ili modernih heuristika, za rešavanje teških optimizacionih problema [92]. Metaheuristike su apstraktne i opšte strategije pretraživanja prostora rešenja koje se uz značajno manji trud mogu prilagoditi rešavanju konkretnog problema. Osnovna karakteristika ovih algoritama je da oni prilikom rešavanja optimizacionih problema postižu zadovoljavajuće rezultate u razumnom vremenskom periodu pri čemu ne zahtevaju ili mogu zahtevati samo nekoliko pretpostavki o datom problemu. Specifično, za rešavanje teških optimizacionih problema kao što su kombinatorni problemi velikih dimenzija i problemi optimizacije nelinearne funkcije sa ograničenjima, metaheuristike često postižu bolji odnos između kvaliteta rezultata i vremena utrošenog za njihovo izračunavanje u odnosu na determinističke pristupe.

Fudamentalne komponente svih metaheuristika su sposobnosti eksploatacije i eksploracije. Eksploatacija podrazumeva sposobnost algoritma da primeni znanje o prethodno pronađenim dobrim rešenjima, dok se eksploracija odnosi na istraživanje novih oblasti prostora pretrage. Eksploatacija i eksploracija su u međusobnoj kontradikciji, tako da je balans ovih sposobnosti od esencijalne važnosti za dobijanje kvalitetnih optimizacionih rezultata.

Velika opštost je dobra karakteristika metaheuristika koja omogućuje njihovu primenu na raznovrstan skup problema. Sa druge strane, njihova primena se ne svodi na jednostavno kodiranje pripremljenog algoritma u željenom programskom jeziku, već zahteva mnogo kreativnog rada. Za većinu metaheuristika, njihova implementacija podrazumeva osmišljavanje prikladne reprezentacije rešenja, definisanje ili modifikovanje operatora pretrage, operatora selekcije i finog podešavanja kontrolnih parametara [100]. Dodatno, kod nekih problema optimizacije nije pogodno vršiti optimizaciju funkcije cilja jer će pretraga uvek rezultovati neodgovarajućim rešenjima. U ovim slučajevima se izvršava neka prosta transformacija funkcije cilja, pri čemu tako dobijenu funkciju nazivamo funkcija podobnosti (eng. fitness function).

Prema načinu kreiranja rešenja optimizacionog problema metaheuristike možemo podeliti u dve klase [100]:

- Metaheuristike bazirane na jednom rešenju (eng. single-based metaheuristics)
- Populacione metaheuristike (eng. population-based metaheuristics)

Klase metaheuristika koje su bazirane na jednom rešenju i mataheuristika koje su bazirane na populacijama se međusobno ne isključuju jer mnogi metaheuristički algoritmi kombinuju ideje iz različitih klasa. Takvi algoritmi se nazivaju hibridne metaheuristike.

Metaheuristike bazirane na jednom rešenju pronalaze nova rešenja iterativno vršeći modifikaciju i unapređivanje jednog tekućeg rešenja s. Algoritam 1 ilustruje osnovne korake ove klase algoritama.

Algorithm 1 Opšti algoritam metaheuristika baziranih na jednom rešenju

Ulaz: Inicijalno rešenje s_0.
t = 1;
repeat
  Generisati ($C(s_t)$);
  s_{t+1} = Selekcija ($C(s_t)$);
  t = t + 1;
until Kriterijum zaustavljanja je zadovoljen
Izlaz: Najbolje pronađeno rešenje.

Početno rešenje se generiše na slučajan način ili pomoću odgovarajuće heuristike. U svakoj iteraciji se smenjuju dve faze: faza generisanja i faza zamene. U fazi generisanja iz tekućeg rešenja s generiše se skup kandidata rešenja $C(s_t)$. Skup $C(s_t)$ se dobija primenom jedne transformacije rešenja s_t. U fazi zamene vrši se selekcija novog rešenja s_{t+1} iz skupa $C(s)$ koje će zameniti postojeće rešenje s_t. Broj iteracija zavisi od nekog unapred definisanog kriterijuma.

Najistaknutije metaheuristike ove klase su metaheuristika lokalne pretrage, tabu pretraga i simulirano kaljenje [100].

Lokalna pretraga (eng. local search) je jedna od najjednostavnijih metaheuristika koja se često koristi kao komponenta složenijih. Ovaj metod je iterativni proces u kom se polazi od nekog početnog rešenja i pretražuje se definisana okolina tog rešenja. Ako je pronađeno bolje rješenje od početnog, postupak se ponavlja u okolini novog rešenja. Ako kriterijum izbora rešenja nije zadovoljan ni za jednog suseda ili je zadovoljen neki drugi kriterijum zaustavljanja metoda staje. Na ovaj način se pronalazi lokalni optimum.

Tabu pretraga (eng. taboo search, TS) je metaheuristika koju je razvio Fred Glover [38], [39]. Ova metaheuristika pripada klasi lokalnih metoda pretrage. Efikasnost lokalne pretrage se povećava korišćenjem memorijske strukture. Memorijska struktura se koristi u cilju obeležavanja nedavno posećenih potencijalnih rešenja tako da algoritam pretrage u nastavku rada više ne posećuje ta rešenja.

Simulirano kaljenje (eng. simulated annealing, SA) je metaheuristika inspirisana postupkom kaljenja metala [56]. Ovaj algoritam iterativno vrši pretragu polazeći od nekog početnog rešenja. U svakoj iteraciji se iz trenutnog rešenja na slučajan nacin bira rešenje koje se nalazi u njegovoj okolini. Ako su ispunjeni određeni uslovi prelazi se u novo rešenje, u suprotnom se ostaje u postojećem rešenju. Način pretrage kontroliše parametar temperature. Što je temperatura viša, veća je verovatnoća za prelazak u lošije rešenje. U početnim iteracijama temperatura je visoka što sprečava zaustavljanje algoritma u lokalnom minimumu. Tokom izvršavanja algoritma temepratura se polako snižava čime se pretraga usmerava ka najboljem rešenju u trenutnoj okolini.

1.5. Populacione metaheuristike

Populacione metaheuristike u iterativnom procesu pretrage stvaraju nova rešenja koristeći više potencijalnih rešenja populacije [100]. Algoritam 2 ilustruje osnovne korake ove klase algoritama.

Algorithm 2 Opšti algoritam populacionih metaheuristika

$P = P_0$.
t = 1;
repeat
  Generisati (P'_t);
  P_{t+1} = Selekcija populacije ($P_t \cup P'_t$);
  t = t + 1;
until Kriterijum zaustavljanja je zadovoljen
Izlaz: Najbolje pronađeno rešenje.

Populaciju čini skup potencijalnih rešenja (jedinki) problema. Populacione metaheuristike vrše iterativna poboljšanja populacije pri čemu su sva potencijalna rešenja deo prostora pretrage. Početna populacija se uglavnom generiše na slučajan način. U svakoj iteraciji smenjuju se faza generisanja

nove populacije P'_t i faza u kojoj će se izvršiti kreiranje populacije P_{t+1} koja će se preneti u sledeću iteraciju.

U fazi generisanja nove populacije P'_t vrše se modifikacije potencijalnih rešenja tekuće populacije pomoću odgovarajućih operatora. Ovi operatori koriste informacije vezane za poznavanje vrednosti funkcije podobnosti jedinka iz populacije. Ponekad se koriste i informacije vezane za istoriju prethodnih stanja jedinki. Operator selekcije se često koristi radi odabira dobrih jedinki koje će učestvovati u formiranju novih rešenja. U cilju kreiranja boljih rešenja koriste se operatori ukrštanja i mutacije. Operatori ukrštanja vrše formiranje novog rešenja tako što kombinuju atribute dva ili više rešenja. Operatori mutacije na slučajan način modifikuju rešenje.

U fazi kreiranja populacije iz tekuće populacije P_t i nove populacije P'_t generiše se populacija P_{t+1} koja će se preneti u sledeću iteraciju. Standardan način na koji se vrši određivanje populacije P_{t+1} jeste prenošenje cele novokreirane populacije P'_t u sledeću iteraciju. Drugi način je da se u ovoj fazi koristi elitistička strategija u cilju pronalaženja najboljih rešenja iz dva skupa P_t i P'_t koja će formirati populaciju P_{t+1}. Na ovaj način se populacija usmerava prema boljim rešenjima prostora pretrage. Broj iteracija zavisi od nekog unapred definisanog kriterijuma.

Pretraga algoritama koji su koncentrisani na unapređivanje samo jednog rešenja dovodi do malih promena u potencijalnom rešenju što ograničava njihovu pretragu na lokalne regione. Stoga se može zaključiti da su metaheuristike iz ove klase orjentisane ka eksploataciji pronađenih dobrih rešenja [100]. Sa druge strane, operatori pretrage populacionih metaheuristika su sofisticirani mehanizmi koji su gotovo bez izuzetka zasnovani na slučajnosti. Iz ovih razloga populacione metaheuristike imaju sposobnost bolje eksploracije celog prostora pretrage što ih čini odgovarajućim izborom za globalnu pretragu.

Grupa populacionih metaheuristika je uglavnom inspirisana prirodom i jedan od glavnih razloga uspešnosti ovih algoritama je to što imitiraju najuspešnije procese u prirodi, koje uključuju biološke sisteme, fizičke i hemijske procese. Populacione metaheuristike čini veliki broj tehnika i mogu se klasifikovati na različite načine u zavisnosti od prirode fenomena koji simuliraju.

Među najstarijim i najpoznatijim tehnikama iz ove grupe algoritama nalaze se evolutivni algoritmi. Evolutivni algoritmi vrše poboljšavanje populacije rešenja iterativno koristeći mehanizme koji su inspirisani evolucijom vrsta. Primeri najpopularnijih evolutivnih algoritama su genetski algoritmi (eng. genetic algorithm, GA) [45], algoritam diferencijalne evolucije (eng.

differential evolution, DE) [94] i algoritam evolucionih strategija (eng. evolution strategy, ES) [9].

Algoritmi inteligencije rojeva su novije metaheuristike koje su inspirisane biološkim sistemima. Ovi algoritmi rade sa populacijom grupa samoorganizovanih jedinki koje se pojavljuju u prirodi. Razmena informacija između jedinki utiče na metodu kreiranja rešenja, čime se simulira inteligentno ponašanje jedinki. Među tehnikama iz ove grupe algoritama nalaze se optimizacija rojevima čestica (eng. particle swarm optimization, PSO) [59], algoritmi inspirisani pčelinjim kolonijama (eng. bees-inspired algorithms) [49, 83, 111], optimizacija mravljim kolonijama (eng. ant colony optimization, ACO) [30], optimizacija populacijom ljudske grupe (eng. seeker optimization algorithm, SOA) [21], optimizacija populacijom bakterija (eng. bacterial foraging optimization, BFO) [19], algoritam svitaca (eng. firefly algorithm, FA) [105], pretraga kukavice (eng. cuckoo search, CS) [108], optimizacija populacijom učenika (eng. teaching-learning-based optimization, TLBO) [74] kao i mnoge druge.

Novi algoritmi inspirisani prirodom, specijalno biološkim sistemima, pojavljuju se skoro svake godine [112]. Takođe, mnogo studija je posvećeno proširivanju i poboljšanju originalnih algoritama, kao i njihovim primenama. Cilj modifikacija originalnih populacionih algoritama je poboljšanje njihovih performansi generalno [28, 32, 33, 34, 35, 70, 82, 95] ili za rešavanje nekih klasa problema [63, 64, 115, 117, 118, 119]. Unapređenja se uglavnom postižu modifikacijom operatora pretrage, operatora selekcije ili finim podešavanjima parametara koji kontrolišu izvršavanje algoritma. Još jedan način za kreiranje poboljšanih varijanti ovih algoritama je uvođenje većih izmena u strukturi i funkcionisanju populacionih algoritama. Ovo se postiže kombinovanjem nekog populacionog algoritma sa nekim determinističkim ili drugim stohastičkim tehnikama.

Sa druge strane, osnovne varijante populacionih metaheuristika su razvijene za rešavanje optimizacionih problema koji imaju samo ograničenja vezana za granice optimizacionih parametara. Međutim, mnogi realni problemi nameću dodatna ograničenja tipa nejednakosti ili jednakosti. Za veliki broj ovih problema teško je generisati dopustiva rešenja. U cilju vođenja pretrage ka dopustivim regionima razvijene su metode za upravljanje ograničenjima. Ove tehnike se integrišu u osnovne varijante populacionih metaheuristika [67].

U daljem tekstu su opisane metode za upravljanje ograničenjima koje se najčešće koriste kod populacionih metaheuristika i detaljan opis populacionih algoritama čija će poboljšanja biti izložena u ovom radu (metaheuris-

tika pčelinjih kolonija, algoritam svitaca, pretraga kukavice i metaheuristika bazirana na populaciji ljudske grupe).

1.5.1. Metode za upravljanje ograničenjima

U početku razvoja i izučavanja populacionih metaheuristika nastao je veliki broj tehnika za upravljanje ograničenjima. Poslednjih godina se izdvojio mali skup tehnika koje su se pokazale kao najefikasnije za rešavanje velikog broja problema [67]. Neke od najpopularnijih tehnika su: kaznene funkcije (eng. penalty functions), pravila bazirana na dopustivosti (eng. feasibility based rules) i stohastičko rangiranje (eng. stohastic ranking, SR).

Kaznene funkcije predstavljaju tehniku za upravljanje ograničenjima koja problem optimizacije sa ograničenjima svodi na problem optimizacije bez ograničenja [90]. U ovom slučaju formula funkcije koju treba optimizovati je:

$$\varphi(x) = f(x) + p(x) \tag{1.2}$$

gde je $f(x)$ funkcija cilja, a $p(x)$ je kaznena vrednost koja se računa pomoću sledeće formule:

$$p(x) = \sum_{i=1}^{q} r_i \cdot max(g_i, 0) + \sum_{j=q+1}^{m} c_j \cdot |h_j(x)| \tag{1.3}$$

gde su r_i i c_j pozitivne konstante koje nazivamo kaznenim koeficijentima.

Postoji više tipova kaznenih funkcija. Najjednostavnija kaznena funkcija je poznata pod nazivom "smrtna kazna". Ovaj pristup nedopustivim rešenjima dodeljuje najgoru moguću vrednost funkcije cilja ili ih jednostavno eliminiše iz procesa optimizacije [8, 99]. Nedostatak korišćenja ovog tipa kaznene funkcije je što se prilikom pretrage ne koriste vredne informacije koje se mogu dobiti iz nedopustivih rešenja.

Sa druge strane postoje tehnike zasnovane na kaznenim funkcijama koje koriste informacije iz nedopustivih rešenja, ali se pored toga fokusiraju i na definisanje odgovarajucih kaznenih faktora. Ove tehnike se razlikuju prema načinu definisanja kaznenih faktora. Najuspešniji primeri ovih tehnika se baziraju na adaptivnim kaznenim funkcijama [40, 41, 75]. Ove funkcije koriste informacije dobijene iz algoritma za ažuriranje vrednosti kaznenih faktora.

Generalno cilj tehnika baziranih na kaznenim funkcijama je favorizacija selekcije dopustivih rešenja, što se postiže smanjenjem vrednosti funkcije cilja nedopustivih rešenja. Prednost ove tehnike je jednostavna implementacija. Međutim, kaznene funkcije zahtevaju fina podešavanja vrednosti kaznenih faktora koje su veoma zavisne od konkretnog problema koji treba optimizovati.

Skup od tri pravila dopustivosti definisao je Deb [27]. Ova pravila se u literaturi često pominju kao Debova pravila. Pravila bazirana na dopustivosti se definišu na sledeći način:

- Kada se porede dva dopustiva rešenja bira se rešenje koje ima bolju vrednost funkcije cilja.
- Kada se porede dopustivo i nedopustivo rešenje bira se dopustivo rešenje.
- Kada se porede dva nedopustiva rešenja bira se rešenje koje ima manju vrednost sume prekršaja ograničenja.

Suma prekršaja ograničenja CV za rešenje x se računa pomoću sledeće formule:

$$CV(x) = \sum_{i=1}^{q} max(g_i, 0) + \sum_{j=q+1}^{m} |h_j(x)| \tag{1.4}$$

Nedostatak ove metode je mogućnost dovođenja algoritma do prevremene konvergencije [68]. Uprkos tome, pravila bazirana na dopustivosti predstavljaju trenutno najpopularniju tehniku za upravljanje ograničenjima. Primena ove tehnike je efikasna i jednostavna. Njena osnovna prednost je činjenica da nema parametre koje je potrebno podešavati.

Stohastičko rangiranje je tehnika razvijena za prevazilaženje nedostataka tehnike bazirane na kaznenim funkcijama [76]. Ovde se misli na nedostatke vezane za izbor kaznenih koeficijenata. Kod tehinke stohastičkog rangiranja umesto kaznenih faktora, uvodi se parametar P_f koji kontroliše kriterijum za upoređivanje nedopustivih rešenja. Za rangiranje rešenja prema vrednosti funkcije cilja, SR koristi algoritam sortiranja koji je sličan bubble sort algoritmu.

Algoritam sortiranja tehnike SR je dat kao Algoritam 3, gde je u je slučajno generisan broj iz opsega (0,1), Num je broj prolaza kroz celu populaciju (ovaj broj bi bio jednak SP u slučaju da upoređivanja nisu izvršena

stohastički), SP je ukupan broj jedinki, I_j je j-ta jedinka populacije, f je funkcija cilja i CV je suma prekršaja ograničenja.

Algorithm 3 Algoritam sortiranja tehnike SR

```
for i = 1 to Num do
  for i = 1 to SP − 1 do
    u = random(0,1)
    if ((CV(I_j) = CV(I_{j+1}) = 0) or (u < P_f)) then
      if (f(I_j) > f(I_{j+1})) then
        zameni(I_j, I_{j+1})
      end if
    else
      if (CV(I_j) > CV(I_{j+1})) then
        zameni(I_j, I_{j+1})
      end if
    end if
  end for
  if (zamena jedinki nije izvršena) then
    break;
  end if
end for
```

U slučaju da je $P_f = 1$ nedopustiva rešenja se sortiraju isključivo prema vrednosti funkcije cilja (eng. over-penalization). Kada je $P_f = 0$ nedopustiva rešenja se sortiraju isključivo prema vrednosti sume prekršaja (eng. under-penalization). Podjednaka šansa za sortiranje nedopustivih rešenja prema jednom od ova dva kriterijuma postiže se za $P_f = 1/2$.

SR tehnika je originalno korišćena kod mehanizma zamene algoritma ES [76]. Ova tehnika je kasnije uspešno kombinovana sa algritmom DE za rešavanje optimizacionih problema sa ograničenjima [69].

1.5.2. Metaheuristika pčelinjih kolonija

Algoritam pčelinjih kolonija (eng artificial bee colony, ABC) koji je razvio Karaboga [49] za rešavanje numeričkih optimizacionih problema, jedan je od najistaknutijih predstavnika populacionih metaheuristika. ABC algoritam simulira proces traganja pčela za hranom. Polovinu populacije pčela čine pčele istraživači, a drugu polovinu populacije čine nezaposlene pčele (posmatrači i izviđači). Ukupan broj izvora hrane jednak je broju pčela istraživača. Svaki izvor hrane je potencijalno rešenje problema. Kvalitet svakog izvora

hrane predstavljen je određenom vrednošću funkcije podobnosti. Pčele istraživači traže hranu oko izvora hrane i u međuvremenu daju informacije o kvalitetu izvora hrane pčelama posmatračima. Pčele posmatrači imaju cilj da odaberu dobre izvore hrane, zavisno od primljenih informacija. Nakon primljenih informacije one dalje tragaju za hranom oko izabranih izvora hrane. Pčele istraživači koje napuštaju nedovoljno dobre izvore hrane da bi istražile nove izvore hrane postaju pčele izviđači.

Opšta struktura ABC algoritma data je kao Algoritam 4.

Algorithm 4 Opšta struktura ABC algoritma

```
t = 1;
Faza inicijalizacije;
repeat
  Istraživačka Faza;
  Posmatračka Faza;
  Izviđačka Faza;
  Memorisati najbolje pronađeno rešenje;
  t = t + 1;
until (t = Maksimalni broj iteracija)
```

U **fazi inicijalizacije** populacija se generiše na slučajan način u prostoru pretrage, vrši se izračunavanje vrednosti funkcije podobnosti generisanih rešenja i izvršava se podešavanje kontrolnih parametara. Vrednost funkcije podobnosti svakog rešenja x izračunava se prema sledećoj formuli:

$$fitness(x) = \begin{cases} 1/(1+f(x)) & , \text{ ako je } f(x) \geq 0 \\ 1+abs(f(x)) & , \text{ ako je } f(x) < 0 \end{cases} \tag{1.5}$$

Osnovni ABC algoritam ima četiri kontrolna parametra: maksimalni broj ciklusa ili iteracija (eng. maximum cycle number, MCN), veličina populacije (eng. size of population, SP) koja predstavlja zbir pčela istraživača i pčela posmatrača, parametar *limit* i broj pčela izviđača. Parametar *limit* predstavlja maksimalni broj pokušaja da se neko rešenje unapredi.

U **istraživačkoj fazi** svako rešenje i, $i = 1, 2, ..., SP/2$, uključeno je u proces ažuriranja koje je dato pomoću sledeće jednačine:

$$v_{ij} = x_{ij} + \varphi_{ij} \cdot (x_{ij} - x_{kj}) \tag{1.6}$$

gde x_{ij} označava j-ti parametar rešenja x_i, j je slučajno izabrani indeks, φ_{ij} je slučajni broj uniformne raspodele iz opsega [-1, 1], a x_j predstavlja

drugo slučajno izabrano rešenje iz populacije. Ovaj proces ažuriranja se završava kada se metod kontrole granica optimizacionih parametara (eng. boundary constraint handling mechanism) primeni za novo rešenje v_i i izvrši pohlepna selekcija (eng. greedy selection) između rešenja x_i i v_i. Metod kontrole granica optimizacionih parametara koji se koristi u osnovnom ABC algoritmu dat je sledećom formulom:

$$v_{ij} = \begin{cases} l_j & \text{, ako je } v_{ij} < l_j \\ u_j & \text{, ako je } v_{ij} > u_j \end{cases} \tag{1.7}$$

U **posmatračkoj fazi** rešenja koja će biti podvrgnuta procesu ažuriranja selektuju se proporcionalno vrednostima funkcije podobnosti. Proces ažuriranja za svako selektovano rešenje u posmatračkoj fazi isti je kao i u istraživačkoj fazi.

U **izviđačkoj fazi**, rešenja koja se nisu menjala u toku izvršenja izvesnog broja ciklusa ponovo se inicijalizuju, odnosno zamenjuju slučajno generisanim novim rešenjima iz prostora pretrage.

Prvu verziju ABC algoritma za rešavanje optimizacionih problema sa ograničenjima predložili su Karaboga i Bastruck u radu [52]. Uvedene su tri modifikacije u odnosu na osnovni ABC algoritam.

Prvo, umesto kriterijuma pohlepne selekcije, ABC algoritam za rešavanje optimizacionih problema sa ograničenjima uključuje Debova pravila [27].

Druga modifikacija je uvođenje novog kontrolnog parametra koji nazivamo rata modifikacije (eng. modification rate - MR). MR parametar kontroliše potencijalne modifikacije optimizacionih parametara u jednačini traženja rešenja ABC algoritma. Iz ovog razloga proces ažuriranja u istraživačkoj i posmatračkoj fazi izvršava se prema sledećoj jednačini:

$$v_{ij} = \begin{cases} x_{ij} + \varphi_{ij} \cdot (x_{ij} - x_{kj}) & \text{, ako je } R_{ij} < MR \\ x_{ij} & \text{, inače} \end{cases} \tag{1.8}$$

gde je φ_{ij} slučajni broj uniformne raspodele u opsegu [-1, 1], x_i predstavlja drugo rešenje izabrano slučajno iz populacije, R_{ij} slučajno izabran realni broj iz opsega (0,1), $j = 1, 2, ..., D$ (D - broj optimizacionih parametara, tj. dimenzija problema).

Treća modifikacija je korišćenje dodatnog kontrolnog parametra u izviđačkoj fazi, nazvanog period produkcije izviđača (eng. scout production period, SPP). Cilj uvođenja ovog parametra je da unapred odredi broj iteracija

nakon kojih će se nedovoljno dobra rešenja zameniti slučajno generisanim novim rešenjima.

Rezultati prve verzije ABC algoritma za rešavanje optimizacionih problema sa ograničenjima [52] dobijeni su testiranjem prvih 13 benčmark funkcija opisanih u radu [79]. Ovi rezultati su upoređeni sa odgovarajućim rezultatima GA i PSO algoritama. Upoređivanje rezultata je pokazalo da ova verzija ABC algoritma može uspešno da se koristi za rešavanje optimizacionih problema sa ograničenjima [52].

Kasnije, u cilju da se poboljšaju rezultati ABC algoritma za rešavanje optimizacionih problema sa ograničenjima, Karaboga i Akay [53] su dodali još dve modifikacije u odnosu na prvu verziju ovog algoritma [52]. Prva modifikacija se odnosi na traženje rešenja prema jednačini 1.8 na sledeći način: u slučaju da se nijedan parametar rešenja ne modifikuje, jedan parametar rešenja se slučajno određuje i modifikuje. Druga promena se odnosi na postupak selekcije za izbor pčela posmatrača. U ovoj varijanti ABC algoritma dopustiva rešenja se selektuju prema verovatnoći proporcionalnoj njihovim vrednostima funkcije podobnosti, dok se nedopustiva rešenja selektuju prema verovatnoći inverzno proporcionalnoj vrednostima njihovih prekoračenja ograničenja. Performanse ove varijante ABC algoritma upoređene su sa performansama devet odgovarajućih algoritama za prvih 13 benčmark funkcija iz rada [79]. Dobijeni rezultati su pokazali da tako predložena varijanta ABC algoritma može efikasno da se koristi za rešavanje optimizacionih problema sa ograničenjima.

Od kako je predložena prva verzija ABC algoritma, razvijeno je mnogo različitih varijanti ABC algoritma koje su primenjene za rešavanje problema iz različitih oblasti [5, 50]. Pored varijanti za rešavanje problema neprekidne optimizacije [1, 2, 13, 51, 52, 53, 60, 65, 66, 78, 91, 101], postoje i verzije ABC algoritma za diskretne i kombinatorne vrste problema [3, 17, 58, 61, 81, 89, 96, 114].

1.5.3. Algoritam svitaca

Algoritam svitaca je populacioni algoritam koji je formulisao je Yang 2008. godine [105]. Ovaj algoritam inspirisan je osobinom svitaca da svetle. Svice odlikuje jačina svetlosti kojom sjaje. Pojava ove svetlosti je delo složenih hemijskih reakcija. Njena osnovna funkcija je atraktivnost kojom privlače partnere ili se štite od predatora. Intenzitet sjaja svitaca smanjuje se kada se udaljenost od svetlećeg objekta povećava [31]. Takođe, sredina absorbuje deo svetlosti kako se udaljenost od svetlećeg objekta povećava. Ove odlike

ponašanja svitaca koristi FA tako da je intenzitet svetlosti proporcionalan funkciji cilja problema koji se optimizuje.

U skladu sa činjenicom da je simulacija prirodnog ponašanja svitaca u algoritmu veoma složena, podrazumevaju se sledeća idealizovana pravila pri konstrukciji FA:

- Svi svici su istog pola.
- Atraktivnost svitaca je proporcionalna jačini svetlosti kojom sjaje.
- Jačina svetlosti svica je pridružena funkciji cilja problema koji se rešava.

Osnovni koraci FA [107] su sledeći:

Korak 1. (Inicijalizacija)

Na početku FA generiše inicijalnu populaciju rešenja na slučajn način, $x_{ik}, i = 1, 2, ..., SP, k = 1, 2, ..., D$. Nakon generisanja inicijalne populacije izračunava se vrednost funkcije cilja za sva rešenja, izvršava se podešavanje kontrolnih parametara i promenljiva t (tekući broj iteracije) setuje se na 1. U osnovnoj verziji FA funkcija podobnosti jednaka je funkciji cilja.

Korak 2. (Računanje nove populacije)

Svako rešenje nove populacije se kreira od rešenja x_i na sledeći način: Za svako rešenje x_i algoritam iterativno proverava svako rešenje x_j , $j = 1, 2, ..., i$, počevši od $j = 1$. Ako rešenje x_j ima nižu vrednost funkcije cilja nego rešenje x_i (svitac x_j je svetliji od svica x_i), vrednost parametra x_{ik}, $k = 1, 2, ..., D$ se ažurira prema sedećoj jednačini:

$$x_{ik} = x_{ik} + \beta \cdot (x_{ik} - x_{jk}) + \alpha \cdot S_k \cdot (rand_k - \frac{1}{2}) \tag{1.9}$$

gde se drugi sabirak odnosi na atraktivnost, a treći se odnosi na randomizaciju.

U drugom sabirku jednačine 1.9 parametar β je atraktivnost svica x_i. U radu [107] izabrana je monotono opadajuća funkcija za opisivanje atraktivnosti svitaca. Odgovarajuća eksponencionalna funkcija data je pomoću sedeće jednačine:

$$\beta = \beta_0 \cdot e^{-\gamma \cdot r_{ij}^2} \tag{1.10}$$

gde je r_{ij} razdaljina između svica x_i i svica x_j, dok su β_0 i γ unapred određeni parametri algoritma, odnosno vrednost maksimalne atraktivnosti i koeficijent absorpcije, respektivno.

Udaljenost r_{ij} između svitaca x_i i x_j dobija se kao Dekartovo rastojanje:

$$r_{ij} = \sqrt{\sum_{k=1}^{D}(x_{ik} - x_{jk})^2} \tag{1.11}$$

Kontrolni parametar β_0 opisuje atraktivnost kada se dva svica nađu u istoj tački prostora pretrage, tj. za njihovu udaljenost $r = 0$. Variranje atraktivnosti sa povećanjem udaljenosti od nekog svica sa kojim se komunicira je određena kontrolnim parametrom γ. Vrednost parametra γ je od suštinske važnosti za brzinu konvergencije i rad algoritma uopšte.

U trećem sabirku jednačine 1.9 $\alpha \in [0, 1]$ je parametar koji određuje slučajnost, S_k su parametri skaliranja, dok je $rand_k$ slučajni broj uniformne raspodele između 0 i 1. Parametar skaliranja S_k se izračunava pomoću sledeće jednačine:

$$S_k = |u_k - l_k| \tag{1.12}$$

Dalje, kad god se vrednost rešenja x_i promeni, FA kontroliše granice optimizacionih parametara kreiranog rešenja i memoriše novu vrednost funkcije cilja umesto prethodne. Postupak kontrole granica optimizacionih parametara se izvršava za svako prihvaćeno rešenje. Mehanizam kontrole granica optimizacionih parametara kod FA je dat pomoću jednačine 1.7.

Poslednje rešenje dobijeno putem jednačine 1.7 je konačno rešenje nove populacije, koje ulazi u sledeći ciklus FA algoritma.

Korak 3. (Redukcija parametara randomizacije)

Kvalitet rešenja se može poboljšati ukoliko se redukuje parametar randomizacije α. Ova redukcija može se obezbediti putem redukcione šeme koja se opisuje jednačinom:

$$\alpha(t) = \alpha(t-1) \cdot \theta^{\frac{1}{MCN}} \tag{1.13}$$

gde je MCN - maksimalni broj ciklusa, t - broj tekuće iteracije, a θ - parametar koji se računa pomoću jednačine:

$$\theta = \frac{10.0^{-4.0}}{0.9} \tag{1.14}$$

Ovaj korak je opcionalan u FA.

Korak 4. (Rangiranje svitaca)

Rangiranje svitaca prema intenzitetu njihove svetlosti, tj. vrednosti funkcije cilja.

Korak 5. (Memorisanje najboljeg rešenja)

Memorisanje najboljeg rešenja pronađenog do sada. Povećavanje promenljive t za 1.

Korak 6. (Ispitivanje kriterijuma kraja)

Ako je vrednost promenljive t jednaka broju maksimalnih iteracija, završiti rad algoritma, inače preći na Korak 2.

U FA postoje tri kontrolna parametra: veličina koraka slučajnog pomeraja α, vrednost atraktivnosti β i koeficijent absorpcije γ. Ostalni kontrolni parametri su broj rešenja SP i maksimalni broj iteracija MCN, koji su zajednički za sve populacione metaheuristike.

Vrednosti kontrolnih parametara u FA mogu da utiču na njegove performanse na različite načine i u različitom obimu [106]. Kontrolni parametar α kontroliše slučajnost ili u proširenom smislu količinu raznovrsnosti rešenja. Uočeno je da se za većinu primena FA njegove performanse mogu poboljšati korišćenjem redukcione šeme koja je opisana jednačinom 1.13. Kontrolni parametar β kontroliše atraktivnost i za većinu primena može se uzeti $\beta_0 = 1$. Kontrolni parametar γ kod većine primena varira od 0.01 do 100. Ovaj parametar može da se poveže sa faktorom skaliranja L (prosečna veličina problema koji se optimizuje). Ako su promene skaliranja značajne, tada se obično setuje $\gamma = 1/\sqrt{L}$, inače se setuje $\gamma = O(1)$. Za veličinu populacije najbolje je uzeti vrednosti od 25 do 40. Vrednosti veće od 50 se ne preporučuju, jer se time znatno povećava vreme izračunavanja [36].

Osnovni FA je primenjen za rešavanje problema neprekidne optimizacije bez ograničenja [105]. Dobijeni rezultati pokazuju da je za ovu klasu problema FA algoritam superiorniji u odnosu na PSO i GA. U radu [36] FA je primenjen za rešavanje problema strukturne optimizacije tako što je za vođenje pretrage ka dopustivim regionima koristio metodu baziranu na kaznenim funkcijama. Rezultati optimizacije su pokazali da je FA efikasniji od drugih metaheuristika, kao što su PSO, GA, DE ili SA.

Za poslednjih pet godina broj istraživača koji usmeravaju pažnju na FA ubrzano raste [106]. Iako je originalni FA predviđen za rešavanje numeričkih problema optimizacije, opisano je dosta njegovih modifikovanih verzija za rešavanje diskretnih i kombinatornih problema. U najnovije vreme FA i njegove varijante nalaze brojne primene u različitim oblastima, kao što su obrada slika, optimizacija problema u industriji, dizajn antena itd.

1.5.4. Pretraga kukavice

Pretraga kukavice je populacioni algoritam koji su razvili Yang i Deb [108] 2009. godine. Ovaj algoritam se zasniva na parazitizmu nekih vrsta kukavica koje polažu svoja jaja u gnezda drugih ptica. CS je podržan takozvanim *Lévy* letom umesto običnog slučajnog kretanja.

U CS algoritmu, gnezdo predstavlja jedno rešenje, dok kukavičje jaje predstavlja novo rešenje. Cilj je koristiti nova i potencijalno bolja rešenja kao zamenu za slabija rešenja u populaciji. U cilju realizacije CS algoritma korišćena su tri idealizovana pravila:

- Svaka kukavica može da položi samo jedno jaje i bira slučajno gnezdo gde jaje polaže.
- Važi princip pohlepne selekcije, tj. samo jaja najboljeg kvaliteta prelaze u sledeću generaciju.
- Broj raspoloživih gnezda je konstantan. Ptica domaćin otkriva jaje kukavice sa verovatnoćom p_a iz segmenta [0,1]. Ako ptica domaćin pronađe kukavičje jaje, može da ga izbaci iz gnezda ili da napusti gnezdo i sagradi novo. Poslednja pretpostavka može se aproksimirati tako što se jedan deo populacije menja na slučajan nacin. Deo populacije koji će biti zamenjen se određuje na osnovu vrednosti parametra p_a.

U osnovnom obliku CS algoritma svako gnezdo sadrži jedno jaje. Ovaj algoritam može da se proširi na složenije slučajeve gde svako gnezdo sadrži više jaja, tj. skup rešenja. CS algoritam ima manje kontrolnih parametara u odnosu na druge populacione metaheuristike koje su poznate u literaturi. Kako su veličina populacije i maksimalni broja iteracija zajednički kontrolni parametri za sve populacione metaheuristike može se zaključiti da CS algoritam ima samo jedan specijalni kontrolni parametar - verovatnoću otkrivanja kukavičjeg jajeta p_a. U većini primena CS algoritma korišćena je vrednost $p_a = 0.25$. Mali broj kontrolnih parametara je značajna prednost CS algoritma u odnosu na ostale populacione metaheuristike jer je podešavanje kontrolnih parametara nekada teži problem od samog optimizacionog problema koji se rešava [4].

Osnovni koraci CS algoritma su sledeći:

Korak 1. (Inicijalizacija)

CS algoritam generiše inicijalnu populaciju rešenja na slučajn način, $x_{ij}, i = 1, 2, ..., SP, j = 1, 2, ..., D$. Nakon generisanja inicijalne populacije

izračunava se vrednost funkcije cilja za sva rešenja, izvršava se podešavanje kontrolnih parametara i promenljiva t (tekući broj iteracije) setuje se na 1. U osnovnoj verziji CS funkcija podobnosti jednaka je funkciji cilja. Pre početka procesa iterativne pretrage CS algoritam utvrđuje najuspešnije rešenje. Ova vrednost se čuva u promenljivoj veličini x_{best}.

Korak 2. (Računanje nove populacije)

Pre početka računanja nove populacije izračunava se korak skaliranja računa pomoću formule:

$$\varphi = \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi \cdot \frac{\beta}{2})}{\Gamma(1+\frac{\beta}{2}) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}} \tag{1.15}$$

gde β označava parametar *Lévy* raspodele, a Γ označava gama funkciju. Za veličinu koraka *Lévy* raspodele koristi se preporučena vrednost parametra $\beta = 1.5$. Na taj način svako rešenje (gnezdo) x_i, $i = 1, 2, ..., SP$ populacije koja vrši pretragu daje novo rešenje v_i koje se računa pomoću jednačine:

$$v_{ij} = x_{ij} + 0.01 \cdot \left(\frac{r_1 \cdot \varphi}{r_2} \right)^{\frac{1}{\beta}} \cdot (x_{ij} - x_{best,j}) \cdot r_3 \tag{1.16}$$

gde su r_1, r_2 i r_3 tri slučajna broja normalne raspodele, $j = 1, 2, ..., D$.

Rešenje x_i se zamenjuje kreiranim rešenjem v_i, ukoliko rešenje v_i ima manju vrednost funkcije cilja (primena pohlepne selekcije). Pri svakom koraku izračunavanja CS algoritam kontroliše granice optimizacionih parametara novih rešenja populacije. Mehanizam kontrole granica optimizacionih parametara kod CS algoritma je dat pomoću jednačine 1.7.

Korak 3. (Memorisanje najboljeg rešenja)

Memorisanje do sada najboljeg pronađenog rešenja, tj. vektora rešenja sa najmanjom vrednošću funkcije cilja.

Korak 4. (Izmene u populaciji na slučajan način)

Za svako rešenje x_i, $i = 1, 2, ..., SN$ primenjuje se operator ukrštanja pomoću jednačine:

$$v_{ij} = \begin{cases} x_{ij} + rand_1 \cdot (x_{permute1,j} - x_{permute2,j}) & , \text{ ako je } rand_2 > p_a \\ x_{ij} & , \text{ inače} \end{cases} \tag{1.17}$$

gde $j = 1, 2, ..., D$. U jednačini 1.17 $rand_1$ i $rand_2$ su slučajni brojevi uniformne raspodele iz opsega $[0, 1]$, respektivno, a $permute_1$ i $permute_2$ su dva

različita niza dobijena pomoću funkcije permutacije koja je primenjena na skup $\{1, 2, ..., SP\}$. Nakon kreiranja novih rešenja, CS algoritam kontroliše granice optimizacionih parametara novih rešenja populacije.

Korak 5. (Memorisanje najboljeg rešenja)

Memorisanje do sada najboljeg pronađenog rešenja i povećavanje promenljive t za 1.

Korak 6. (Ispitivanje kriterijuma kraja)

Ako je vrednost promenljive t jednaka broju maksimalnih iteracija, završiti rad algoritma, inače preći na Korak 2.

Osnovni CS algoritam primenjen je za rešavanje problema neprekidne optimizacije bez ograničenja [108]. Rezultati optimizacije pokazuju da je za ovu klasu problema CS algoritam superiorniji u odnosu na PSO i GA. Ovaj algoritam je do sada uspešno primenjen za rešavanje više teških optimizacionih problema kao što su optimizacija strukturnih inženjerskih problema [37], [109], procena uloženog napora pri testiranju softvera [93], problem bojenja planarnog grafa [121], raspored permutacija toka prodaje [62], itd.

1.5.5. Metaheuristika bazirana na populaciji ljudske grupe

Metaheuristika bazirana na populaciji ljudske grupe (eng. seeker optimization algorithm, SOA) simulira ponašanje ljudi prilikom pretraživanja koje je bazirano na njihovoj memoriji, iskustvu, nesigurnom rezonovanju i međusobnoj komunikaciji [22].

Osnovni koraci algoritma pretrage ljudske grupe su prikazani Algoritmom 5.

Algorithm 5 Opšta struktura algoritma pretraživanja ljudske grupe

```
t = 1;
Inicijalizacija populacije;
repeat
    Izračunati pravac pretrage i dužinu koraka svakog tragača;
    Ažurirati pozicije svakog tragača;
    Izvršiti evaluaciju i memorisati najbolje pozicije svakog tragača;
    Implementirati operaciju internog učenja u podpopulaciji;
    Memorisati najbolje pronađeno rešenje;
    t = t + 1;
until (t = Maksimalni broj iteracija)
```

Populacija algoritma pretraživanja ljudske grupe je organizovana u tri jednake podgrupe ili podpopulacije prema indeksima jedinki. Zbog naziva algoritma, jedinke populacije ovog algoritma nazivamo tragači. Svi tragači koji pripadaju istoj podgrupi čine okolinu koje predstavlja socijalnu komponentu za socijalno deljenje informacija.

Inicijalna populacija rešenja (jedinki ili tragača), $x_{ij}, i = 1, 2, ..., SP$, $j = 1, 2, ..., D$ generiše se na slučajan način u prostoru pretrage. SP je ukupan broj jedinki ili tragača, a D je dimenzija problema. Proizvoljni tragač x_i ima sledeće karakteristike: tekuću poziciju $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$, ličnu najbolju poziciju p_{ibest} i najbolju poziciju u svojoj okolini g_{best}. Nakon kreiranja inicijalne populacije izračunava se vrednost funkcije cilja za sva rešenja, tj. izvršava se evaluacija kreiranih rešenja. Pre početka procesa iterativne pretrage algoritam utvrđuje najuspešnije rešenje populacije i za svakog tragača vektor lične najbolje pozicije i vektor najbolje pozicije u njegovoj okolini.

Najvažnije karakteristike procesa pretrage algoritma pretraživanja ljudske grupe su:

- Algoritam koristi pravac pretrage i dužinu koraka za ažuriranje pozicija tragača.
- Pravac pretrage se računa na osnovu kompromisa između egoističnog, altruističnog i proaktivnog ponašanja.
- Za računanje dužine koraka koristi se fazi rezonovanje zbog nesigurnog rezonovanja prilikom procesa ljudskog traženja. Pravilo nesigurnog rezonovanja inteligentnog traženja se može opisati na sledeći način: "Ako je vrednost funkcije cilja mala, onda je dužina koraka mala".

Pravac pretrage je određen pomoću egoističnog, altruističnog i proaktivnog ponašanja tragača. Ponašanje tragača se smatra egoističkim ako on veruje da treba da se kreće ka svojoj ličnoj najboljoj poziciji kroz kognitivno učenje. Tragači koji se ponašaju altruistički žele da komuniciraju sa ostalim tragačima i da modifikuju svoje ponašanje na osnovu odgovora drugih tragača koji se nalaze u njihovoj okolini. Ako tragač hoće da promeni svoj pravac pretrage i da preusmeri svoje ponašanje prema svom ponašanju iz prošlosti, onda smatramo da je njegovo ponašanje proaktivno.

Formula po kojoj se izračunava pravac pretrage d_{ij}, $j = 1, 2, ..., D$, koji modeluje ove tipove ponašanja, za i-tog tragača je:

$$d_{ij} = sign(w \cdot (p_{ibest_j} - x_{ij}) + r_1 \cdot (g_{best_j} - x_{ij}) + \\ + r_2 \cdot (x_{ij}(t_1) - x_{ij}(t_2))) \tag{1.18}$$

gde je funkcija $sign()$ funkcija znaka (eng. sign function), parametar w je težina inercije (eng. inertia weight), $t_1, t_2 \in \{t,\ t-1,\ t-2\}$, $x(t_1)$ i $x(t_2)$ predstavljaju najbolju i najlošiju poziciju u skupu $\{x(t),\ x(t-1),\ x(t-2)\}$ redom, r_1 i r_2 su realni brojevi koji su izabrani na slučajan način iz opsega [0,1]. Balans između globalne eksploracije i lokalne eksploatacije se obezbeđuje linearnim smanjivanjem vrednosti parametra w od vrednosti 0.9 do vrednosti 0.1.

Dužina koraka određuje se korišćenjem fazi sistema. Vrednosti funkcije cilja svake podpopulacije sortirane su u opadajućem redosledu, pri čemu su im dodeljeni redni brojevi od 1 do SS. Redni brojevi se uzimaju kao ulazne vrednosti za fazi rezonovanje. Stepen pripadnosti (eng. membership degree) tragača x_i se daje pomoću jednakosti:

$$\mu_i = \mu_{max} - \frac{SS - I_i}{SS - 1} \cdot (\mu_{max} - \mu_{min}) \tag{1.19}$$

gde SS označava veličinu podpopulacije kojoj i-ti tragač pripada, I_i je redni broj vektora koji odgovara tragaču x_i nakon sortiranja vrednosti funkcije cilja u opadajućem redosledu, μ_{max} je maksimalni stepen pripadnosti i obično ima vrednost oko 1.0.

Fazi sistem koristi princip kontrolnog pravila: "*ako* (uslovni deo) *onda je* (akcioni deo)". Funkcija pripadnosti "bell" je korišćena za akcioni deo:

$$\mu(x) = e^{\frac{-x^2}{2\delta^2}} \tag{1.20}$$

Zbog jednostavnosti, posmatra se samo jedna promenljiva. Kada se koristi "bell"-ova funkcija pripadnosti, vrednosti stepena pripadnosti ulaznih varijabli koje su izvan opsega $[-3\delta, 3\delta]$ su manje od 0.011. Iz ovog razloga je za minimalni stepen pripadnosti, μ_{min}, uzeta vrednost 0.011.

Parametar δ_i "bell"-ove funkcije pripadnosti se za i-tog tragača računa pomoću jednakosti:

$$\delta_{ij} = w \cdot |x_{best_j} - x_{avg_j}|,\ j = 1, 2, ..., D \tag{1.21}$$

gde je x_{best} pozicija najboljeg tragača u podpopulaciji kojoj pripada i-ti tragač, x_{avg} je vektor čije su vrednosti prosečne vrednosti pozicija tragača koji pripadaju istoj podpopulaciji.

U cilju uvođenja slučajnosti radi unapređenja sposobnosti lokalne pretrage, koristi se sledeća jednačina:

$$\mu_{ij} = rand(\mu_i, 1),\ j = 1, 2, ..., D \tag{1.22}$$

gde je $rand(\mu_i, 1)$ slučajni broj iz opsega $[\mu_i, 1]$.

Jednačina koja se koristi za generisanje dužine koraka i-tog tragača je:

$$\alpha_{ij} = \delta_{ij} \cdot \sqrt{-\ln(\mu_{ij})},\ j = 1, 2, ..., D \tag{1.23}$$

Pravac pretrage α_{ij} i dužina koraka d_{ij} se odvojeno izračunavaju za svaku jedinku x_i i za svako $j = 1, 2, ..., D$ u svakoj iteraciji t pomoću jednačine:

$$x_{ij}(t+1) = x_{ij}(t) + \alpha_{ij}(t) \cdot d_{ij} \tag{1.24}$$

gde je $\alpha_{ij} \geq 0$ i $d_{ij} \in \{-1, 0, 1\}$.

Jednačina 1.24 je osnovna jednačina pretrage SOA.

U svakoj iteraciji, tekuće pozicije dve najlošije jedinke iz svake populacije se zamenjuju dvema najboljim jedinkama iz druge dve podpopulacije. Ovaj proces se naziva **učenje između podpopulacija**. Ukoliko bi proces pretrage bio vođen samo na osnovu lokalnih informacija, došlo bi do prevremene konvergencije i algoritam bi se zaglavio u lokalnom minimumu. Faza učenja između podpopulacija omogućava razmenu informacija između podpopulacija i posledično izbegavanje prevremene konvergencije algoritma.

SOA je razvijen za rešavanje problema neprekidne optimizacije [21] i analiziran na izazovnom broju benčmark problema. U radu [22] njegove performanse su upoređene sa performansama DE algoritma i tri modifikovane verzije algoritma PSO. Pri ovom poređenju SOA je pokazao mogućnost bolje globalne pretrage, posebno za jednomodalne funkcije. Za multimodalne funkcije rezultati nisu bili posebno zadovoljavajući, jer je uočeno za ove probleme algoritam može jednostavno da se zaglavi u nekom od lokalnih optimuma.

Od kako je predstavljen, SOA je uspešno primenjen za rešavanje različitih optimizacionih problema. U radu [23] SOA je primenjen za podešavanje strukture i parametara veštaških neuronskih mreža. SOA je u radu [25] primenjen za optimizaciju razmene energije kroz ćelijsku membranu, dok je u radu [24] evolucioni metod baziran na ovom algoritmu predložen za dizajn digitalnog IIR (eng. infinite impulse response) filtra. Algoritam pretraživanja ljudske grupe je u radu [26] preimenovan u algoritam optimizacije

ljudskom grupom (eng. human group optimizer, HGO), jer ovaj naziv bolje ukazuje ne suštinu ovog algoritma.

Glava 2

Algoritam ABC baziran na operatoru ukrštanja

Od nastanka ABC algoritma predloženo je nekoliko uspešnih poboljšanih varijanti ovog algoritma za rešavanje problema neprekidne optimizacije sa ograničenjima.

Menzura-Montes i Cetina-Dominguez [65] predložili su varijantu ABC algoritma koja modifikuje ponašanje pčela izviđača (eng. scout-behavior modified artificial bee colony, SM-ABC). U SM-ABC algoritmu ponašanje pčela izviđača se menja korišćenjem novog operatora koji povećava sposobnost algoritma da generiše rešenja koja se nalaze u regiji prostora pretrage koja obećava dobre izvore hrane, odnosno rezultate. Dalje, SM-ABC uvodi mehanizam dinamičkog smanjenja tolerancije u cilju lakšeg rešavanja ograničenja tipa jednakosti. Predloženi SM-ABC algoritam testiran je na prvih 13 benčmark funkcija iz rada [79]. Dobijeni rezultati su pokazali da SM-ABC ima bolje performanse od ABC algoritma [53].

Imajući u vidu ideje korišćene u radu [65], isti autori su kasnije uveli još dve modifikacije u SM-ABC algoritam: turnirsku selekciju umesto selekcije zasnovane na verovatnoći proporcionalnoj vrednostima funkcije podobnosti. Druga modifikacija je novi mehanizam kontrole granica optimizacionih parametara [66]. Ovaj novi algoritam, nazvan modifikovani ABC (eng. modified artificial bee colony, M-ABC), je prva verzija ABC algoritma koja je testirana na 24 benčmark funkcije koje su opisane u radu [79]. Upoređivanje konačnih rezultata je pokazalo da M-ABC algoritam daje značajno bolja najbolja rešenja i prosečne vrednosti u odnosu na ABC algoritam za rešavanje optimizacionih problema sa ograničenjima.

Nedavno, uspešnu varijantu ABC algoritma za rešavanje optimizacionih

problema sa ograničenjima predložili su Li i Yin [78]. Ovaj algoritam nazvan je samoadaptirajući ABC algoritam (eng. self-adaptive constrained artificial bee colony, SACABC). Modifikacije uvedene u SACABC algoritmu dovode do znatnih promena strukture i funkcionisanja ABC algoritma. U izviđačkoj i posmatračkoj fazi ovog algoritma koriste se dva različita, nova operatora pretrage koja su inspirisna operatorima pretrage DE algoritma. Podešavanje MR parametra izvršava se samoadaptirajućim postupkom. Ovaj algoritam, kao i M-ABC algoritam, koristi novi mehanizam kontrole granica optimizacionih parametara. Konačno, SACABC algoritam nema izviđačku fazu i koristi dve različite metode za upravljanje ograničenjima, Debova pravila i metod koji se koristi kod multiobjektne optimizacije. SACABC je testiran na 24 benčmark funkcije koje su opisane u radu [79]. Dobijeni rezultati su pokazali da ovaj algoritam postiže visoko kvalitetne rezultate u odnosu na ABC [53], M-ABC [66] i druge odgovarajuće algoritme, u većini slučajeva.

U ovoj glavi disertacije uvodi se nova varijanta ABC algoritma sa ciljem daljih poboljšanja njegovih sposobnosti za rešavanje optimizacionih problema sa ograničenjima. Uvedeni algoritam nazvan je algoritam pčelinjih kolonija baziran na operatoru ukrštanja (eng. crossover-based artificial bee colony algorithm, CB-ABC). Iako se ABC algoritam uspešno koristi u mnogim oblastima, u radu [50] je primećeno da korišćenje operatora ukrštanja može da poboljša brzinu konvergencije ka globalnom optimumu. Iz ovog razloga CB-ABC algoritam koristi uniformni operator ukrštanja u izviđačkoj fazi, umesto slučajnog pretraživanja. Dalje, u radu [122] je primećeno da operator pretrage ABC algoritma može biti manje efikasan u eksploataciji postojećih rešenja. U cilju prevazilaženja uočenih nedostataka ABC algoritma korišćena su dva modifikovana operatora pretrage u istraživačkoj i posmatračkoj fazi. Uvedene su još dve modifikacije koje sa odnose na korišćenje dinamičkog smanjenja tolerancije kod ograničenja oblika jednakosti i korišćenje unapređene metode za ograničenja koja se odnose na granične vrednosti optimizacionih parametara.

Predloženi algoritam je testiran na velikom skupu od 24 standardne benčmark funkcije i 4 problema inženjerskog dizajna. Dobijeni rezultati su pokazali da CB-ABC algoritam postiže kvalitetnije rezultate, da je robustniji i da ima bržu konvergenciju u odnosu na varijante ABC algoritma koje su razvijene za rešavanje optimizacionih problema sa ograničenjima. Dodatno, CB-ABC algoritam ima bolje performanse u većini slučajeva u poređenju sa drugim odgovarajućim metaheuristikama.

CB-ABC algoritam detaljno je predstavljen u Odeljku 2.1. U Odeljku 2.2 su izloženi benčmark problemi, objašnjena su setovanja parametara i

izvršena je analiza postignutih rezultata.

2.1. Opis predloženog algoritma: CB-ABC

U skladu sa dosadašnjim radom na poboljšanju ABC algoritma, CB-ABC algoritam uvodi pet modifikacija u odnosu na prvu varijantu ABC algoritma koja je predložena za rešavanje problema neprekidne optimizacije sa ograničenjima u radu [52]. Dve modifikacije sa odnose na korišćenje dinamičkog smanjenja tolerancije kod ograničenja oblika jednakosti i korišćenje unapređene metode za ograničenja koja se odnose na granične vrednosti optimizacionih parametara.

Algoritam CB-ABC koristi mehanizam kontrole ograničenja koja se odnose na granične vrednosti optimizacionih parametara (eng. boundary constraint-handling mechanism) koji je baziran na radu Kukkonena i Lampinena [57]. Ovaj mehanizam je opisan jednačinom:

$$v_{ij} = \begin{cases} 2 \cdot l_j - v_{ij} & \text{, ako je } v_{ij} < l_j \\ 2 \cdot u_j - v_{ij} & \text{, ako je } v_{ij} > u_j \\ v_{ij} & \text{, inače} \end{cases} \tag{2.1}$$

gde je v_{ij} j-ti parametar potencijalnog rešenja v_i, a l_j i u_j donja i gornja granica optimizacionog parametra v_{ij}, respektivno.

Korišćenjem ovog mehanizma u slučaju da novokreirano rešenje prekorači granice nekih optimizacionih parametara, generiše se različiti skup vrednosti ovih parametara. Na taj način se postiže održavanje raznovrsnosti populacije. Dakle, ukoliko ima mnogo rešenja fokusiranih ka ekstremnim vrednostima prostora pretrage, ovaj način vraćanja vrednosti parametara u prostor pretrage pomaže algoritmu da izbegne zaglavljivanje u lokalnom minimumu.

Standardan način rešavanja ograničenja tipa jednakosti jeste njihova transformacija u ograničenja tipa nejednakosti. Dakle, svako ograničenje tipa jednakosti $h_j(x) = 0$ zameniće se sledećim ograničenjima tipa nejednakosti: $|h_j(x)| - \epsilon \leq 0$, gde je $\epsilon > 0$ dozvoljena tolerancija.

Za dozvoljenu toleranciju ϵ se uzima veoma mali broj i kod ABC algoritma vrednost tolerancije se ne menja tokom pretrage [52]. Ovaj način traženja dopustivog rešenja koji zadovoljava ograničenja tipa jednakosti nije uvek najpogodniji, jer je dopustiv region veoma mali u poređenju sa prostorom pretrage. Stoga pronalaženje bilo kog dopustivog rešenja može da bude teško i optimizacioni proces može da bude narušen [70].

Sa druge strane, postoje različiti mehanizmi kontrole ograničenja tipa jednakosti koji dinamički menjaju vrednost tolerancije tokom procesa optimizacije [66]. CB-ABC algoritam koristi jedan ovakav pristup koji je inicijalno predložen za algoritam ES [40]. Prema ovom pristupu vrednost tolerancije se definiše pomoću jednačine:

$$\epsilon = \begin{cases} 1 & \text{, ako je } t = 1 \\ e^{-(E(t)\cdot dec)/E_f} & \text{, ako je } 1 < t < s \\ \epsilon_{min} & \text{, ako je } t \geq s \end{cases} \tag{2.2}$$

gde je t - broj tekuće iteracije, $E(t)$ - broj tekuće evaluacije, E_f je ukupan broj evaluacija koje algoritam treba da izvrši, ϵ_{min} je konačna vrednost tolerancije, a s je broj evaluacije u kojoj korisnik želi da ϵ postane ϵ_{min}. Vrednost parametra s mora da bude manja ili jednaka od vrednosti E_f. Inače, broj koji kontroliše brzinu smanjivanja tolerancije *dec* definiše se na sledeći način:

$$dec = N \cdot \frac{E_f}{s} \tag{2.3}$$

gde je N vrednost eksponenta korišćena da se uspostavi vrednost tolerancije ϵ_{min}, tj. $e^{-N} = \epsilon_{min}$.

Prema jednačini 2.2, u početnim iteracijama procesa optimizacije, u cilju dobijanja nekih dopustivih rešenja, CB-ABC algoritam koristi veću vrednost tolerancije. Na ovaj način se privremeno povećava dopustivi skup rešenja. Kako pretraga napreduje kroz iteracije, vrednost tolerancije se smanjuje. Manja vrednost tolerancije forsira algoritam da prethodno pronađena dopustiva rešenja poboljša, tj. da ih uklopi u novi smanjeni dopustivi skup rešenja.

U ovom poglavlju su dalje opisane tri modifikacije koje se odnose na operatore pretrage koji se koriste u istraživačkoj, posmatračkoj i izviđačkoj fazi, sa ciljem da se poboljša sposobnost eksploatacije ABC algoritma.

2.1.1. Modifikacije u istraživačkoj fazi

U cilju generisanja novog rešenja v_i, u istraživačkoj fazi CB-ABC algoritam koristi sledeću jednačinu pretrage:

$$v_{ij} = \begin{cases} x_{ij} + \varphi_i \cdot (x_{ij} - x_{kj}) & \text{, ako je } R_{ij} < MR \\ x_{ij} & \text{, inače} \end{cases} \tag{2.4}$$

gde je φ_i slučajni broj uniformne raspodele iz opsega [-1,1], x_k je slučajno selektovano rešenje iz populacije koje je različito od rešenja x_i, R_{ij} je slučajno izabran realni broj iz opsega [0,1] i $j = 1, 2, ..., D$.

U originalnom ABC algoritmu za rešavanje optimizacionih problema sa ograničenjima prilikom modifikacije svakog parametra x_{ij} koriste se različiti slučajni brojevi φ_{ij} [52]. Sa druge strane, prema jednačini 2.4 prilikom modifikacije svakog parametra x_{ij}, φ_i je isti slučajni broj iz opsega [-1,1]. Na ovaj način smanjen je prostor u kome se može generisati novo rešenje v_{ij}.

Kontrolni parametar MR ima značajnu ulogu u kontroli balansa eksploatacije i eksploracije ABC algoritma [2]. Vrednost MR parametra je u opsegu (0,1] i kod originalnog ABC algoritma za vreme procesa pretrage vrednost ovog parametra je fiksna [52]. Niže vrednosti MR parametra dovode do promena manjeg broja optimizacionih parametara, što ima za posledicu sporiju brzinu konvergencije [2]. Sa druge strane, različiti problemi traže različite vrednosti odnosa eksploatacije i eksploracije u toku procesa pretrage. Iz tog razloga korišćenje veće konstantne vrednosti parametra MR može da dovede to toga da se algoritam zaglavi u lokalnom optimumu.

Sa namerom da se obezbedi odgovarajući balans eksploatacije/eksploracije, CB-ABC u početnim iteracijama povećava vrednost parametra MR od vrednosti 0.1 do unapred definisane vrednosti MR_{max}, dok se vrednost $MR = MR_{max}$ koristi u ostalim iteracijama. Zato se na kraju svake iteracije vrednost MR parametra ažurira na sledeći način:

$$MR = \begin{cases} MR + \frac{(MR_{max} - 0.1)}{P \cdot MCN} & \text{, ako je } MR < MR_{max} \\ MR_{max} & \text{, inače} \end{cases} \tag{2.5}$$

Maksimalni broj ciklusa ili iteracija (MCN) je zajednički kontrolni parametar za sve populacione algoritame. Ovaj parametar se određuje empirijski i predstavlja dovoljan broj iteracija tokom kojih bi algoritam trebalo da konvergira ka nekom rešenju. Kontrolni parametar P je novi parametar uključen u cilju održavanja raznovrsnosti populacije i, posledično, kontrole balansa eksploatacije/eksploracije. Ovaj parametar određuje broj početnih iteracija CB-ABC algoritma tokom kojih će se vrednost MR parametra iterativno povećavati. Vrednost kontrolnog parametra P je iz opsega [0,1].

2.1.2. Modifikacije u posmatračkoj fazi

U posmatračkoj fazi CB-ABC algoritma rešenja koja će biti uključena u proces pretrage selektuju se prema verovatnoći koja je data jednačinom:

$$P_i = 0.9 \cdot (fit_i/maxfit) + 0.1 \tag{2.6}$$

gde je *maxfit* najbolja vrednost funkcije podobnosti rešenja iz populacije, a fit_i označava vrednost funkcije podobnosti rešenja x_i [12, 78].

U ovoj fazi CB-ABC algoritam uvodi novu jednačinu pretrage:

$$v_{ij} = x_{ij} + \varphi_i \cdot (x_{lj} - x_{kj}) \tag{2.7}$$

gde je φ_i slučajni broj uniformne raspodele iz opsega [-1, 1], x_l i x_k predstavljaju druga dva rešenja slučajno selektovana iz populacije i $j = 1, 2, ..., D$. Prema jednačini 2.7 prostor gde se generiše novo rešenje v_i je proširen u odnosu na jednačinu 2.4. Dodatno, prema jednačini 2.7 svaki parametar x_{ij} rešenja x_i se menja u cilju kreiranja rešenja v_i što utiče na povećanje raznovrsnosti populacije.

2.1.3. Modifikacije u izviđačkoj fazi

U cilju da se poboljša razmena dobrih informacija u populaciji, predloženi CB-ABC algoritam koristi uniformni operator ukrštanja u izviđačkoj fazi, na sledeći način:

Posle svake SPP-te iteracije, svako rešenje koje se nije promenilo najmanje *limit* broj puta, zamenjuje se novim rešenjem koje se kreira prema jednačini:

$$v_{ij} = \begin{cases} y_j & , \text{ ako je } R_{ij} < 0.5 \\ x_{ij} & , \text{ inače} \end{cases} \tag{2.8}$$

gde je y_j j-ti element najboljeg rešenja pronađenog do tog trenutka, R_{ij} je slučajno izabran realni broj iz opsega [0,1] i $j = 1, 2, ..., D$.

2.1.4. Pseudo-kod CB-ABC algoritma

Pseudo-kod predloženog algoritma CB-ABC dat je Algoritmom 6.

Algorithm 6 Pseudo-kod CB-ABC algoritma

Inicijalizovati kontrolne parametre SP, MCN, P i $limit$
Inicijalizovati populaciju rešenja na slučajan način u prostoru pretrage x_{ij}, $i = 1 \ldots, SP/2$, $j = 1 \ldots, D$
Izvršiti evaluaciju svakog rešenja x_i, $i = 1 \ldots, SP/2$
$MR = 0.1$
$t = 1$
if (postoje ograničenja tipa jednakosti) **then**
  Inicijalizacija $\epsilon(t)$
end if
repeat
  for $i = 1$ to $SP/2$ **do**
    Kreirati novo rešenje v_i za svako postojeće rešenje x_i koristeći jednačinu 2.4
    Izvršiti kontrolu graničnih vrednosti parametara kreiranog rešenja v_i koristeći jednačinu 2.1 i izvršiti evaluaciju ovog rešenja
    Primeniti proces selekcije baziran na Debovim pravilima
  end for
  for $i = 1$ to $SP/2$ **do**
    Izračunati vrednost verovatnoće P_i za svako rešenje x_i koristeći jednačinu 2.6
  end for
  t = 1
  i = 1
  while $(t \leq SP/2)$ **do**
    if $(\delta_i < P_i)$ **then**
      $t = t + 1$
      Kreirati novo rešenje v_i za svako izabrano rešenje x_i koristeći jednačinu 2.7
      Izvršiti kontrolu graničnih vrednosti parametara kreiranog rešenja v_i koristeći jednačinu 2.1 i izvršiti evaluaciju ovog rešenja
      Primeniti proces selekcije baziran na Debovim pravilima
    end if
    $i = i + 1$
    if (vrednost i je jednaka $SP/2$) **then**
      $i = 1$
    end if
  end while

if (t mod $SPP = 0$) **then**
Svako rešenje koje se nije poboljšalo najmanje *limit* broj puta se zamenjuje novim rešenjem koje se kreira pomoću jednačine 2.8
end if
Ažurirati vrednost MR parametra koristeći jednačinu 2.5
Memorisati najbolje pronađeno rešenje
$t = t + 1$
if (postoje ograničenja tipa jednakosti) **then**
Ažurirati vrednost $\epsilon(t)$ koristeći jednačinu 2.2
end if
until $t = MCN$

Važno je uočiti razliku između kontrolnih parametara koje koriste CB-ABC i ABC algoritmi, jer podešavanje kontrolnih parametara nekog algoritma može biti složenije nego sam problem koji se rešava [4]. Oba algoritma koriste SP i MCN parametre, koji su opšti kontrolni parametri za sve prirodom inspirisane algoritme. Od specijalnih kontrolnih parametara oba algoritma koriste MR, SPP i *limit* kontrolne parametre. Jedina razlika je u tome što CB-ABC algoritam koristi MR kontrolni parametar samo u istraživačkoj fazi i što je taj parametar u ovoj fazi podeljen na dva parametra, MR_{max} i P.

2.1.5. Balans eksploatacije i eksploracije kod CB-ABC

Uspešnost pretrage svakog populacionog algoritma zavisi od dobro uspostavljenog odnosa eksploatacije i eksploracije. Raznovrsnost populacije odnosi se na razlike među jedinkama populacije i predstavlja značajni faktor za određivanje balansa eksploracije i eksploatacije [103]. Glavna pretpostavka leži u činjenici da je veća raznovrsnost populacije neophodna kako bi algoritam izbegao lokalne optimume, dok manja raznovrsnost omogućava da konačni rezultati budu što bliži globalnom optimumu.

U originalnom ABC algoritmu faktori koji imaju direktan uticaj na sposobnosti eksploracije i eksploatacije su jednačina pretrage i mehanizam selekcije. Kako ni jedan operator pretrage ne utiče isključivo na jednu od ove dve sposobnosti [103], može se zaključiti da se kod ABC algoritma procesi eksploatacije i eksploracije prepliću u toku istraživačke i posmatračke faze. U izviđačkoj fazi izvršava se samo eksploracija produkcijom slučajnih novih rešenja u prostoru pretrage.

Prema jednačini 1.8 originalnog ABC algoritma za rešavanje optimizacionih problema sa ograničenjima, koeficijent φ_j je slučajni broj u opsegu

[-1,1] generisan za svaki optimizacioni parametar x_{ij} koji se menja, a x_k je slučajno izabrano rešenje populacije koje je različito od rešenja x_i. Iz ovih činjenica se može zaključiti da je jednacina 1.8 zasnovana na prevelikoj slučajnosti što je dobro za eksploraciju prostora pretrage. Uzimajući u obzir činjenicu da su eksploatacija i eksploracija međusobno kontradiktorne sposobnosti, dolazimo do zaključka da jednačina 1.8 ne obezbeđuje dovoljno dobru eksploataciju pronađenih rešenja. Takođe, imajući u vidu da izviđačka faza obezbeđuje samo eksploraciju prostora pretrage, može se zaključiti da je ekploatacija pronađenih rešenja kod ABC algoritma za rešavanje optimizacionih problema sa ograničenjima nedovoljno dobra. U cilju da obezbedi bolji balans eksploatacije i eksploracije, CB-ABC algoritam koristi drugu jednačinu pretrage u istraživačkoj fazi, uvodi novi operatop pretrage u posmatračkoj fazi i operator ukrštanja u izviđačkoj fazi.

U istraživačkoj fazi CB-ABC algoritma, raznovrsnost populacije je smanjena smanjenjem prostora gde mogu da se generišu nova potencijalna rešenja. Ovo se postiže korišenjem istog slučajnog broja φ iz opsega [-1,1] za svaki optimizacioni parametar x_{ij} koji se menja pomoću jednačine 2.4. Dodatno, empirijski je utvrđeno da korišćenje jednačine 2.4 zahteva višu vrednost MR parametra, kako bi se osigurala eksploracija prostora pretrage. Ipak, korišćenjem nižih vrednosti MR parametra u početnim iteracijama smanjuje se verovatnoća propuštanja dobrih rešenja, kao i nemogućnost dovoljne eksploatacije pronađenih rešenja. Na ovaj način se performanse algoritma za rešavanje nekih optimizacionih problema sa ograničenjima mogu značajno poboljšati. Sa druge strane, kako se MCN vrednost može izabrati da bude dovoljno velika da osigura konvergenciju algoritma, ova modifikacija ne utiče na performanse algoritma za druge probleme koji zahtevaju konstantnu veću vrednost MR parametra tokom pretrage.

U posmatračkoj fazi, raznovrsnost populacije se povećava menjanjem svakog optimizacionog parametra potencijalnog rešenja. Takođe, korišćenjem jednačine 2.7, prostor gde se novo rešenje može generisati se povećava u odnosu na prostor u kome se generišu rešenja u istraživačkoj fazi CB-ABC algoritma. Iz ovog sledi da posmatračka faza CB-ABC algoritma obezbeđuje bolju eksploraciju dobrih rešenja, u poređenju sa istraživačkom fazom CB-ABC algoritma.

U izvidjačkoj fazi se koristi uniformni operator ukrštanja sa ciljem da se poveća verovatnoća nalaženja novih regiona pretrage sa dobrim rešenjima.

Može se primetiti da svaki od tri operatora pretrage koji se koriste u CB-ABC algoritmu proizvode raznovrsnost populacije. Ali u poređenju sa ABC algoritmom za rešavanje optimizacionih problema sa ograničenjima, jasno je

da se ova raznovrsnost redukuje u istraživačkoj fazi i u izviđačkoj fazi. Iz toga sledi da u ovim fazama CB-ABC algoritam ima poboljšanu sposobnost eksploatacije u odnosu na ABC algoritam. Pored toga, korišenjem različitih jednačina pretrage u istraživačkoj i u posmatračkoj fazi, CB-ABC algoritam obezbeđuje bolje pretraživanje okoline kvalitetnih rešenja. Iz toga se može zaključiti da je kod CB-ABC algoritma uspostavljen bolji balans eksploatacije i eksploracije između ove dve faze.

2.2. Eksperimentalni rezultati

Predloženi CB-ABC algoritam testiran je na skupu od 24 benčmark problema i četiri optimizaciona problema inženjerskog dizajna. Ovaj algoritam implementiran je u programskom jeziku Java. Dobijeni rezultati su upoređeni sa odgovarajućim rezultatima algoritama ABC [53], M-ABC [66], SACABC [78] i najuspešnijim algoritmima koji su prethodno upoređivani sa ABC algoritmom. Rezultati svih pomenutih metaheuristika su uzeti iz odgovarajuće literature.

2.2.1. Benčmark funkcije

Za testiranje performansi CB-ABC algoritma korišćen je skup benčmark funkcija koje su namenjene za konkurentno rešavanje optimizacionih problema sa ograničenjima. Matematički modeli ovih funkcija mogu da se nađu u tehničkom izveštaju CEC' 2006 [79]. Ovaj skup funkcija sadrži različite tipove funkcija cilja, kao što su linearne, kvadratne i nelinearne. Osnovne karakteristike ovih funkcija date su u Tabeli 2.1, gde je D broj optimizacionih parametara, $\rho(\%)$ je odnos između dopustivog regiona i celog prostora pretrage, LN je broj linearnih ograničenja tipa nejednakosti, NN je broj nelinearnih ograničenja tipa nejednakosti, LJ je broj linearnih ograničenja tipa jednakosti, NJ je broj nelinearnih ograničenja tipa jednakosti, a je broj aktivnih ograničenja, dok je $f(x^*)$ vrednost funkcije cilja najboljeg poznatog rešenja.

2.2.2. Podešavanje parametara

CB-ABC algoritam u svim eksperimentima koristi sledeće vrednosti kontrolnih parametara:

- Veličina populacije: $SP = 90$;

Tabela 2.1: Osnovne karakteristike benčmark funkcija

Prob.	D	Vrsta funkcije	$\rho(\%)$	LN	NN	LJ	NJ	a	$f(x^*)$
g01	13	kvadratna	0.0003	9	0	0	0	6	-15.000
g02	20	nelinearna	99.9962	1	1	0	0	1	-0.8036191
g03	10	nelinearna	0.0002	0	0	0	1	1	-1.000
g04	5	kvadratna	26.9089	0	6	0	0	2	-30665.539
g05	4	nelinearna	0.0000	2	0	0	3	3	5126.497
g06	2	nelinearna	0.0065	0	2	0	0	2	-6961.814
g07	10	kvadratna	0.0001	3	5	0	0	6	24.306
g08	2	nelinearna	0.8488	0	2	0	0	0	-0.095825
g09	7	nelinearna	0.5319	0	4	0	0	2	680.630
g10	8	linearna	0.0005	3	3	0	0	6	7049.248
g11	2	kvadratna	0.0099	0	0	0	1	1	0.7499
g12	3	kvadratna	4.7452	0	9	0	0	0	-1.000
g13	5	nelinearna	0.0000	0	0	1	2	3	0.053942
g14	10	nelinearna	0.0000	0	0	3	0	3	-47.765
g15	3	kvadratna	0.0000	0	0	1	1	2	961.715
g16	5	nelinearna	0.0204	4	34	0	0	4	-1.905
g17	6	nelinearna	0.0000	0	0	4	4	4	8853.539675
g18	9	kvadratna	0.0000	0	13	0	0	6	-0.866025
g19	15	nelinearna	33.4761	0	5	0	0	0	32.656
g20	24	linearna	0.0000	0	6	2	12	16	0.0967
g21	7	linearna	0.0000	0	1	0	5	6	193.725
g22	22	linearna	0.0000	0	1	8	11	19	236.431
g23	9	linearna	0.0000	0	2	3	1	6	-400.055
g24	2	linearna	79.6556	0	2	0	0	2	-5.508

- Procenat inicijalnih iteracija u kojima se MR parametar dinamički ažurira : $P = 0.3$;
- Konačna vrednost parametra: $MR_{max} = 0.9$;
- Period produkcije pčela izviđača: $SPP = 350$.
- Broj pokušaja unapređenja nekog izvora hrane pre njegovog ažuriranja: $limit = 1$.

U cilju poboljšanja kvaliteta rezultata CB-ABC algoritam koristiti nešto veće vrednosti SP i MR kontrolnih parametara od vrednosti ovih parametara koje su korišćene u originalnom ABC algoritmu [52, 53]. Vrednosti P, SPP i $limit$ kontrolnih parametara empirijski su utvrđene.

Pri svakom izvršavanju CB-ABC algoritma, MCN za pet benčmark problema (g14, g17, g20, g22 i g23) je 2660. Za ove funkcije maksimalni

broj evaluacija funkcije podobnosti (eng. number of function evaluations, FES) korišćen u istraživačkoj i u posmatračkoj fazi je $90 \cdot 2660 = 239400$, dok je u izviđačkoj fazi $FES = 315$. Zbog toga je za ovih pet benčmark problema broj evaluacija 239715. Ova cena izračunavanja se može smatrati jednakom ceni izračunavanja state-of-the art algoritama gde je $FES = 240000$. Za četiri benčmark funkcije (g02, g13, g19 i g21) MCN je 2200. Za šest benčmark funkcija (g01, g05, g07, g10, g15 i g18) MCN je 1500. Za probleme g03 i g11 vrednost MCN je 1000, dok je za g04, g06, g09 i g16 vrednost $MCN = 500$. MCN je 100 za problem g08, 150 za problem g12 i 300 za problem g24.

CB-ABC algoritam koristi sledeće vrednosti parametara mehanizma kontrole ograničenja tipa jednakosti:

- Inicijalna vrednost tolerancije: $\epsilon = 1$;
- Konačna vrednost tolerancije: $\epsilon_{min} = 0.0001$ [79];
- Broj evaluacija kada korisnik želi da ϵ postane ϵ_{min}: S = $0.75 \cdot FES$.

Da bi rezultati različitih algoritama bili uporedivi, za svaku benčmark funkciju, ponovljeno je 30 nezavisnih izvršavanja CB-ABC algoritma.

2.2.3. Generalne performanse CB-ABC algoritma

U Tabeli 2.2 prikazane su najbolje vrednosti, srednje vrednosti i standardne devijacije dobijene tokom 30 nezavisnih izvršavanja CB-ABC algoritma za 24 benčmark funkcije. U ovoj tabeli je još prikazan i maksimalni broj evaluacija potreban za dobijanje ovih rezultata. Najbolja vrednost funkcije cilja pokazuje sposobnost algoritma da postigne optimalni rezultat. Prosečne vrednosti i standardne devijacije određuju robustnost ili stabilnost algoritma. Maksimalni broj evaluacija može da se posmatra kao brzina konvergencije ili cena izračunavanja.

Kao što se može videti iz Tabele 2.2, predloženi CB-ABC algoritam je pronašao optimalna ili najbolja poznata rešenja za 22 od 24 benčmark problema. Za probleme g20 i g22 nisu pronađena dopustiva rešenja. Za 19 problema (g01, g03, g04, g05, g06, g07, g08, g09, g10, g11, g12, g14, g15, g16, g18, g19, g21, g23 and g24) CB-ABC algoritam je dobio najbolja poznata rešenja ili rešenja veoma bliska najboljim poznatim rešenjima pri svakom od 30 izvršavanja. Za probleme g02, g13 i g17 globalni optimum nije dobjen pri svakom izvršavanju CB-ABC algoritma.

Benčmark problem g02 je poznat kao težak za rešavanje, jer ima veoma neravnomerno raspoređene regione pretrage [77]. Za funkciju g13, CB-ABC algoritam nalazi optimalnu vrednost skoro pri svakom izvršavanju. Ipak u jednom od 30 izvršavanja CB-ABC algoritam se zaglavljuje u lokalnom minimumu kome odgovara vrednost funkcije cilja 0.43880.

Za benčmark funkciju g17, CB-ABC algoritam je pronašao bolje od do sada najboljeg poznatog rešenja [79]. U radu [79] optimalno rešenje za g17 je 8853.53967480648, a dozvoljena vrednost tolerancije 0.0001. Iz toga sledi da je poboljšanje sigurno, imajući u vidu da je korišćena ista vrednost tolerancije. U Tabeli 2.3 za problem g17 prikazane su vrednosti prethodnog najboljeg rešenja, novo najbolje rešenje dobijeno algoritmom CB-ABC i odgovarajući parametri novog rešenja.

2.2.4. Upoređivanje CB-ABC sa ABC, SR, ISR i OPA

Imajući u vidu da je CB-ABC algoritam poboljšana varijanta ABC algoritma za rešavanje optimizacionih problema sa ograničenjima [53], u ovoj Sekciji su upoređene performanse CB-ABC i ABC algoritama. Kako su u radu [53] prikazani rezultati za samo 13 benčmark funkcija iz rada [79], upoređivanje je izvršeno za te probleme.

U radu [53] rezultati ABC algoritma upoređeni su sa devet odgovarajućih algoritama. Od tih devet algoritama, poboljšana varijanta stohastičkog rangiranja (eng. improved stohastic ranking, ISR) [77] za većinu testiranih funkcija dobija bolje rezultate od rezultata ABC algoritma. Algoritmi stohastičko rangiranje (eng. stohastic ranking, SR) [76] i "over penalty" pristup (eng. over penalty approch, OPA) [77] dobili su slične ili nešto slabije rezultate u odnosu na rezultate dobijene ABC algoritmom. Iz tog razloga su i algoritmi SR, ISR i OPA uključeni u poređenje performansi sa algoritmom CB-ABC.

Rezultati ABC algoritma dobijeni su za 240000 evaluacija, dok je broj evaluacija 350,000 kod algoritama SR, ISR i OPA. U ABC algoritmu korišćena je konstantna vrednost tolerancije $\epsilon = 0.001$, dok su drugi algoritmi koristili niže vrednosti tolerancije [53]. Broj nezavisnih izvršavanja ovih algoritma je 30 [53].

Tabele 2.4 i 2.5 prikazuju najbolja i prosečna rešenja dobijena pomoću SR, ISR, OPA, ABC i CB-ABC algoritama. Najbolja rešenja su istaknuta (eng. boldface). Na osnovu rezultata iz ovih Tabela jasno se uočava da CB-ABC algoritam uglavnom postiže bolje ili jednake rezultate za sve testirane funkcije u poređenju sa ostalim algoritmima.

Tabela 2.2: Statistički rezultati dobijeni pomoću CB-ABC algoritma za 24 test funkcije za 30 nezavisnih pokretanja. "NF" znači da dopustivo rešenje nije pronađeno.

Prob.	Najbolje rešenje	Prosečno rešenje	Najlošije rešenje	Standardna devijacija	Broj evaluacija
g01	-15.000	-15.000	-15.000	5.03E-15	135,180
g02	-0.8036191	-0.7945223	-0.777844	8.32E-03	198,270
g03	-1.0005	-1.0005	-1.0005	3.64E-07	90,090
g04	-30665.539	-30665.539	-30665.539	8.72E-11	45,045
g05	5126.497	5126.497	5126.497	1.07E-10	135,180
g06	-6961.814	-6961.814	-6961.814	1.82E-12	45,045
g07	24.3062	24.3062	24.3062	4.16E-07	135,180
g08	-0.095825	-0.095825	-0.095825	2.87E-17	8000
g09	680.630	680.630	680.630	2.77E-09	45,045
g10	7049.248	7049.248	7049.248	3.98E-05	135,180
g11	0.7499	0.7499	0.7499	1.29E-10	90,090
g12	-1.000	-1.000	-1.000	0.00E-00	13,500
g13	0.053942	0.066770	0.43880	6.91E-02	198,270
g14	-47.7649	-47.7649	-47.7648	1.02E-05	239,715
g15	961.715	961.715	961.715	2.81E-11	135,180
g16	-1.905	-1.905	-1.905	7.90E-11	45,045
g17	8853.533875	8902.869928	8941.940741	3.74E+01	239,715
g18	-0.866025	-0.866025	-0.866025	1.72E-08	135,180
g19	32.6556	32.6556	32.6557	1.88E-05	198,270
g20	NF	NF	NF	NF	239,715
g21	193.725	193.725	193.725	2.17E-06	198,270
g22	NF	NF	NF	NF	239,715
g23	-400.055	-400.055	-400.055	6.89E-05	239,715
g24	-5.508	-5.508	-5.508	7.15E-15	27,000

Tabela 2.3: Bolje rešenje od najboljeg rešenja datog u radu [79]

Prob.	Prethodno rešenje $f(x)$	Novo rešenje $f(x^*)$	Vrednosti parametara novog rešenja
g017	8853.539675	8853.533875	201.78446249354963 99.99999999999994 383.07103485277105 419.9999999999984 -10.907681379950336 0.07314823120842924

Tabela 2.4: Najbolje rešenje dobijeno algoritmima SR [76], ISR [77], OPA [77], ABC [53] i CB-ABC za 13 test funkcija

Prob.	SR	ISR	OPA	ABC	CB-ABC
g01	-15.000	-15.000	-15.000	-15.000	-15.000
g02	-0.803515	**-0.803619**	**-0.803619**	-0.803598	**-0.8036191**
g03	**-1.000**	**-1.001**	-0.747	**-1.000**	**-1.0005**
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	**5126.497**	**5126.497**	**5126.497**	5126.484	**5126.4967**
g06	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**
g07	24.307	**24.306**	**24.306**	24.330	**24.3062**
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	**680.630**	**680.630**	**680.630**	680.634	**680.630**
g10	7559.192	7049.250	**7049.248**	7053.904	**7049.248**
g11	0.750	0.750	0.750	0.750	**0.7499**
g12	-1.000	-1.000	-1.000	-1.000	-1.000
g13	0.053957	**0.053942**	0.447118	0.760	**0.053942**

Iz Tabele 2.4 se može primetiti da je CB-ABC algoritam pronašao pet boljih najboljih rezultata u odnosu na SR (g02, g07, g10, g11 i g13), dva bolja najbolja rezultata u odnosu na ISR (g10 i g11), tri bolja najbolja rezultata u odnosu na OPA (g03, g11 i g13) i sedam boljih najboljih rezultata u odnosu na ABC (g02, g05, g07, g09, g10, g11 i g13). Specijalno, iz Tabele 2.4 vidimo da je ABC algoritam ostvario nižu vrednost najboljeg rezultata za problem g05 od vrednosti dobijene algoritmom CB-ABC. Međutim, rezultati CB-ABC pristupa su bolji jer problem g05 ima ograničenja tipa jednakosti za čije rešavanje CB-ABC algoritam koristi nižu vrednost tolerancije.

Iz Tabele 2.5 može se videti da CB-ABC algoritam daje bolje ili jednake prosečne vrednosti od ostalih algoritama, sa izuzetkom problema g13, gde CB-ABC pokazuje neznatno manju stabilnost od SR algoritma. CB-ABC postiže sedam boljih prosečnih vrednosti u odnosu na SR (g02, g05, g06, g07, g09, g10 i g11), dve bolje prosečne vrednosti u odnosu na SR (g02 i g11), sedam boljih prosečnih vrednosti u odnosu na OPA (g02, g03, g05, g07, g11, g12 i g13) i osam boljih prosečnih rezultata u odnosu na ABC (g02, g05, g06, g07, g09, g10, g11 i g13).

Prema rezultatima datim u Tabelama 2.4 i 2.5 očigledno je da predloženi CB-ABC algoritam prevazilazi rezultate svih ovih algoritama u odnosu na kvalitet rešenja, robustnost i brzinu konvergencije, za većinu testiranih funkcija. Dalje, u radu [53] rezultati postignuti ABC pristupom su upoređeni sa rezultatima još šest odgovarajućih algoritama: metodom homomorfnog

Tabela 2.5: Prosečno rešenje dobijeno algoritmima SR [76], ISR [77], OPA [77], ABC [53] i CB-ABC za 13 test funkcija

Prob.	SR	ISR	OPA	ABC	CB-ABC
g01	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**
g02	-0.781975	-0.782725	-0.776283	-0.792412	**-0.7945223**
g03	**-1.000**	**-1.001**	-0.257	**-1.000**	**-1.0005**
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5128.881	**5126.497**	5268.610	5185.714	**5126.497**
g06	-6875.940	**-6961.814**	**-6961.814**	-6961.813	**-6961.814**
g07	24.374	**24.306**	24.307	24.473	**24.3062**
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.656	**680.630**	**680.630**	680.640	**680.630**
g10	7559.192	7049.250	**7049.248**	7224.407	**7049.248**
g11	0.750	0.750	0.756	0.750	**0.7499**
g12	**-1.000**	**-1.000**	-0.9998899	**-1.000**	**-1.000**
g13	**0.057006**	0.066770	0.964323	0.968	0.066770

mapiranja (eng. homomorphous mapping, HM), dve varijante evolucionih algoritama, genetskim algoritmom, algoritmom diferencijalne evolucuje i varijante metaheuristike bazirane na rojevima čestica za rešavanje optimizacionih problema sa ograničenjima. Imajući u vidu da rezultati iz Tabele 2.4 i Tabele 2.5 jasno pokazuju da CB-ABC ima bolje performanse od ABC algoritma, ovaj zaključak takođe važi i za ovih šest metaheuristika.

2.2.5. Upoređivanje CB-ABC sa M-ABC i SCABC

U ovoj Sekciji, rezultati predloženog CB-ABC algoritma upoređeni su sa rezultatima algoritama M-ABC [66] i SACABC [78]. Rezultati M-ABC i SACABC algoritama dobijeni su za 240000 evaluacija. M-ABC i SACABC algoritmi koriste istu vrednost tolerancije $\epsilon = 0.0001$ za ograničenja tipa jednakosti, kao i algoritam CB-ABC. Broj nezavisnih izvršavanja M-ABC algoritma je 30, dok je za SACABC ova vrednost 25. Kako je CB-ABC algoritam izvršavan po 30 puta, važno je napomenuti da se pri radu ABC algoritma vrednosti najboljih rešenja dobijaju bar jednom za 10 izvršavanja, a za ostale benčmark probleme skoro pri svakom nezavisnom izvršavanju, tako da je upoređivanje CB-ABC i SACABC regularno izvršeno.

Tabela 2.6: Uporedni rezultati dobijeni pomoću algoritama M-ABC [66] i CB-ABC za 24 test funkcije. "NF" znači da dopustivo rešenje nije pronađeno.

Prob.	Najbolje rešenje		Prosečno rešenje		Najlošije rešenje	
	M-ABC	CB-ABC	M-ABC	CB-ABC	M-ABC	CB-ABC
g01	-15	-15.000	-15	-15.000	-15	-15.000
g02	-0.803615	**-0.8036191**	**-0.799336**	-0.7945223	-0.777438	**−0.777844**
g03	1.000	-1.0005	-1.000	-1.0005	-1.000	-1.0005
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.736	**5126.497**	5178.139	**5126.497**	5317.196	**5126.497**
g06	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
g07	24.315	**24.3062**	24.415	**24.3062**	24.854	**24.3062**
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.632	**680.630**	680.647	**680.630**	680.691	**680.630**
g10	7051.706	**7049.248**	7233.882	**7049.248**	7473.109	**7049.248**
g11	0.75	**0.7499**	0.75	**0.7499**	0.75	**0.7499**
g12	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
g13	0.053985	**0.053942**	0.158552	**0.066770**	0.442905	0.43880
g14	-47.641	**-47.7649**	-47.271	**-47.7649**	-46.537	**-47.7648**
g15	961.715	961.715	961.719	**961.715**	961.793	**961.715**
g16	-1.905	-1.905	-1.905	-1.905	- 1.905	-1.905
g17	8866.618	**8853.533875**	8987.459	**8902.869928**	9165.219	**8941.940741**
g18	-0.866006	**-0.866025**	-0.7950187	**-0.866025**	-0.672216	**-0.866025**
g19	33.285	**32.6556**	34.267	**32.6556**	35.746	**32.6557**
g20	NF	NF	NF	NF	NF	NF
g21	266.500	**193.725**	306.609	**193.725**	329.960	**193.725**
g22	NF	NF	NF	NF	NF	NF
g23	-159.739	**-400.055**	-35.272	**-400.055**	109.010	**-400.055**
g24	-5.508	-5.508	-5.508	-5.508	-5.508	-5.508

Tabela 2.7: Uporedni rezultati dobijeni pomoću algoritama SACABC [78] i CB-ABC za 24 test funkcije. "NF" znači da dopustivo rešenje nije pronađeno.

Prob.	Najbolje rešenje		Prosečno rešenje		Najlošije rešenje	
	SACABC	CB-ABC	SACABC	CB-ABC	SACABC	CB-ABC
g01	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000
g02	-0.803618	**-0.8036191**	-0.7886	**-0.7945223**	-0.7688	**-0.777844**
g03	-1.0005	-1.0005	-1.0004	**-1.0005**	-0.9997	**-1.0005**
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.497	5126.497	5126.497	5126.497	5126.497	5126.497
g06	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
g07	24.3062	24.3062	24.3062	24.3062	24.3064	**24.3062**
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.630	680.630	680.630	680.630	680.630	680.630
g10	7049.248	7049.248	7049.248	7049.248	7049.248	7049.248
g11	0.7499	0.7499	0.7499	0.7499	0.7499	0.7499
g12	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
g13	0.05394	0.053942	0.09242	**0.066770**	0.438802	0.43880
g14	-47.7648	**-47.7649**	-47.7648	**-47.7649**	-47.7648	-47.7648
g15	961.715	961.715	961.715	961.715	961.715	961.715
g16	-1.905	-1.905	-1.905	-1.905	-1.905	-1.905
g17	8,853.534	8853.533875	**8853.631**	8902.869928	**8854.504**	8941.940741
g18	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025
g19	32.6556	32.6556	32.6560	**32.6556**	32.6597	**32.6557**
g20	NF	NF	NF	NF	NF	NF
g21	193.725	193.725	233.0180	**193.725**	324.702	**193.725**
g22	NF	NF	NF	NF	NF	NF
g23	-400.055	-400.055	-400.055	-400.055	-400.055	-400.055
g24	-5.508	-5.508	-5.508	-5.508	-5.508	-5.508

Upoređivanje performansi CB-ABC u odnosu na M-ABC može se videti u Tabelama 2.6, dok su u Tabeli 2.7 dati rezultati uporedjivanja CB-ABC i SACABC algoritama. Najbolji rezultati su istaknuti. U ovim Tabelama nijedan rezultat nije istaknut u slucaju da oba algoritma imaju jednake rezultate. Problemi g20 i g22 su isključeni iz diskusije, jer za njih ni jedan od ovih algoritama nije dao dopustiva rešenja.

Iz Tabele 2.6 se uočava da CB-ABC postiže bolje najbolje rezultate od algoritma M-ABC za trinaest problema (g02, g05, g07, g09, g10, g11, g13, g14, g17, g18, g19, g21 i g23), a jednake najbolje rezultate na devet ostalih funkcija. U odnosu na prosečne vrednosti i najslabije dobijene rezultate, CB-ABC algoritam je bolji od M-ABC pristupa za trinaest problema (g05, g07, g09, g10, g11, g13, g14, g15, g17, g18, g21 i g23). Za problem g02, CB-ABC dobija nešto slabiju prosečnu vrednost i sličan najslabiji rezultat. Za ostalih osam problema srednje vrednosti i najslabiji rezultati su jednaki. Može se zakljuciti da su, osim za problem g02, rezultati dobijeni algoritmom CB-ABC značajno bolji u odnosu na rezultate M-ABC algoritma.

Prema rezultatima izloženim u Tabeli 2.7 jasno je da algoritam SACABC postiže visoko kvalitetne rezultate za 22 benčmark problema. U odnosu na SACABC može se videti da CB-ABC nalazi bolje najbolje rezultate za probleme g02 i g14 i jednake najbolje rezultate za ostale probleme.

U odnosu na prosečne vrednosti i najslabije rezultate, CB-ABC algoritam pokazuje bolje performanse za sedam problema (g02, g03, g07, g13, g14, g19 i g21), dok su rezultati algoritma SACABC za problem g17 stabilniji. Specijalno, iz Tabele 2.7 se može zaključiti da za problem g03 CB-ABC algoritam dobija rezultate za oko 60% manji broj evaluacija. Za problem g07 cena izračunavanja je manja za 45%. Za probleme g02, g13 i g21 smanjenje cene izračunavanja je oko 17%. Za ostalih četrnaest problema SACABC i CB-ABC postižu jednake prosečne i najslabije vrednosti, s tim da je za CB-ABC većina rezultata dobijena za znatno manji broj evaluacija.

Može se zaključiti da rezultati poređenja CB-ABC i SACABC metaheuristika potvrđuju da CB-ABC ima bolje performanse u odnosu na kvalitet dobijenih rešenja, robustnost i brzinu konvergencije, za većinu problema. Dodatno, SACABC algoritam je u radu [78] upoređen sa dva druga odgovarajuća algoritma koja su uspešno korišćena za rešavanje optimizacionih problema sa ograničenjima. To su hibridni algoritam evolucije (eng. hybrid constrained optimization evolutionary algorithm, HCOEA) i adaptivni kompromisni model evolucionih strategija (eng. adaptive trade-off model evolutionary strategy, ATMES). Ova dva algoritma postižu visoko kvalitetne rezultate za prvih 13 funkcija [79]. U radu [78] je zaključeno da je SACABC

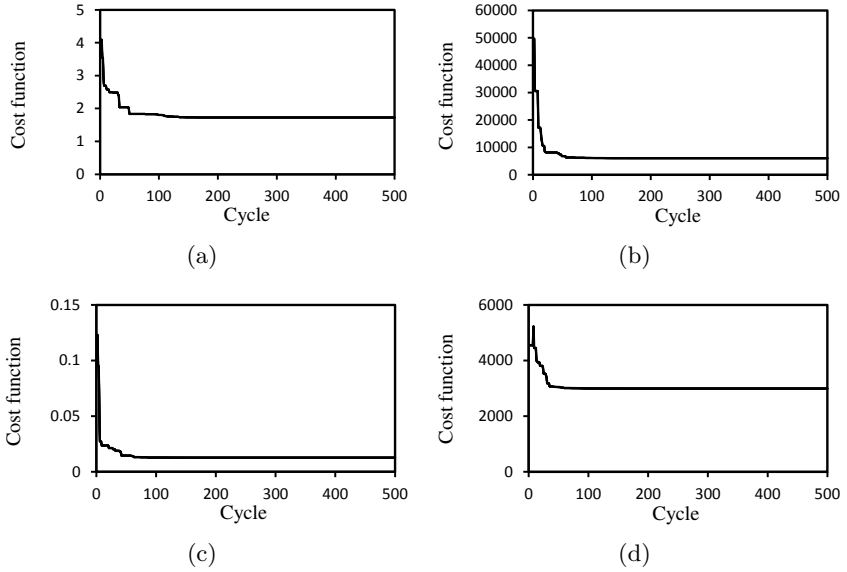
dobio bolje rezultate pri poređenju sa ATMES i HCOEA algoritmima. Iz ovog sledi da se ovaj zaključak takođe odnosi i na predloženi algoritam CB-ABC, imajući u vidu da su za ove test probleme njegove performanse bolje od SACABC algoritma.

2.2.6. CB-ABC za rešavanje problema inženjerskog dizajna

U ovom delu predloženi CB-ABC algoritam je testiran za rešavanje realnih problema sa namerom da se bolje razjasni njegova efikasnost. Rešavana su četiri poznata problema inženjerskog dizajna: zavarena greda (eng. welded beam), posuda pod pritiskom (eng. pressure vessel), pritisna/kompresiona opruga (eng. tension/compression spring) i reduktor brzine (eng. speed reducer). Opisi i matematički modeli ovih problema dati su u radu [1]. Rezultati predloženog CB-ABC algoritma upoređeni su sa rezultatima koji su dobijeni korišćenjem ABC algoritma proširenim za rešavanje inženjerskih optimizacionih problema[1].

Tabela 2.8: Uporedni rezultati dobijeni pomoću ABC [1] i CB-ABC algoritama za rešavanje četiri standardna inženjerska problema.

Prob.	Statstika	ABC	CB-ABC
Zavarena greda	Najbolje rešenje	1.724852	1.724852
	Prosečno rešenje	1.741913	**1.724852**
	St. dev.	3.1E-02	**2.38E-11**
	Broj evaluacija	30,000	**15,000**
Posuda pod pritiskom	Najbolje rešenje	6059.714736	**6059.714335**
	Prosečno rešenje	6245.308144	**6126.623676**
	St. dev.	2.05E+02	**1.14E+02**
	Broj evaluacija	30,000	**15,000**
Pritisna/ kompresiona opruga	Najbolje rešenje	0.012665	0.012665
	Prosečna vrednost	0.012709	**0.012671**
	St. dev.	0.012813	**1.42E-05**
	Broj evaluacija	30,000	**15,000**
Reduktor brzine	Najbolje rešenje	2997.058412	**2994.471066**
	Prosečno rešenje	2997.058412	**2994.471066**
	St. dev.	0	2.48E-07
	Broj evaluacija	30,000	**15,000**



Slika 2.1: Kriva konvergencije najboljeg rezultata dobijenog pomoću CB-ABC algoritma za 30 izvršavanja: (a) Zavarena greda, (b) Posuda pod pritiskom, (c) Pritisna/kompresiona opruga, (d) Reduktor brzine.

U CB-ABC algoritmu korišena je vrednost 30 za veličinu populacije. Za sva četiri inženjerska problema rezultati su dobijeni za 15000 evaluacija. Vrednosti specifičnih kontrolnih parametara CB-ABC algoritma setovane su isto kako je dato u Sekciji 2.2.2.. Rezultati ABC algoritma [1] dobijeni su za 30000 evaluacija. Statistički rezultati ABC i CB-ABC algoritama dobijeni su za po 30 nezavisnih izvršavanja, za svaki problem.

Kako problemi posuda pod pritiskom i reduktor brzine imaju i neprekidne i diskretne promenljive, treba napomenuti da su vrednosti diskretnih promenljivih u CB-ABC algoritmu pronađene tako što su realne vrednosti zaokruživane na najbliže diskretne vrednosti.

U Tabeli 2.8 prikazane su vrednosti najboljih rešenja, prosečnih rešenja i vrednosti standardnih devijacija dobijene CB-ABC i ABC algoritmima, kao i maksimalni broj evaluacija funkcije podobnosti.

Prema Tabeli 2.8 jasno je da CB-ABC postiže bolje ili jednake najbolje rezultate od ABC algoritma [1], za sve testirane inženjerske probleme. Dalje, za svaki testirani problem inženjerskog dizajna najbolji rezultat dobijen CB-ABC algoritmom jednak je do sada najboljem objavljenom zaokruženom rezultatu [63, 110]. U odnosu na prosečne vrednosti i vrednosti standardnih

devijacija, takođe se uočava da CB-ABC algoritam ima bolje performanse od ABC algoritma, za sve testirane probleme. Osim toga, CB-ABC algoritam dolazi do tih rezultata za dva puta manju vrednost broja evaluacija nego ABC algoritam.

Dijagrami konvergencije koji odgovaraju najboljim rešenjima koji su dobijeni nakon 30 nezavisnih pokretanja CB-ABC algoritma dati su na Slici 2.1. Kako se može videti sa Slike 2.1, algoritam CB-ABC veoma brzo konvergira ka optimalnom rešenju, u ranim iteracijama, za svaki od četiri problema inženjerskog dizajna.

Glava 3

Hibridni algoritam pretraživanja ljudske grupe

U ovom delu je izložen hibridni algoritam pretraživanja ljudske grupe (eng. hybrid seeker optimization algorithm, HSO) za rešavanje problema globalne optimizacije. U radu [22] je uočeno da algoritam pretraživanja ljudske grupe ima tendenciju zaglavljivanja u lokalnom minimumu kod multimodalnih funkcija. Sa druge strane u radovima [54, 55] je pokazano da se metaheuristika pčelinjih kolonija veoma uspešno koristi za probleme globalne optimizacije. U cilju poboljšanja performansi SOA za optimizaciju multimodalnih funkcija, kreiran je hibridni algoritam koji uključuje elemente ABC algoritma u algoritam pretraživanja ljudske grupe.

Predloženi hibridni algoritam modifikuje algoritam pretraživanja ljudske grupe, tako što njegovu jednačinu pretrage kombinuje sa dvema jednačinama pretrage metaheuristike pčelinjih kolonija. Takođe, predloženi algoritam koristi uniformni operator ukrštanja u fazi učenja između podpopulacija. HSO algoritam je testiran na kompletnom skupu od 23 standardne benčmark funkcije. Uporedni rezultati pokazuju da predloženi algoritam nadmašuje performanse algoritma pretraživanja ljudske grupe, metaheuristike pčelinjih kolonija, kao i drugih state-of-the-art algoritama u odnosu na kvalitet dobijenih rešenja i robustnost u većini testiranih slučajeva.

HSO algoritam predstavljen je u Odeljku 3.1. U Odeljku 3.2 su izložene karakteristike benčmark problema, dok je u Odeljku 3.3 izvršena analiza postignutih rezultata.

3.1. Predloženi HSO algoritam

Hibridni algoritam pretraživanja ljudske grupe u poređenju sa algoritmom pretraživanja ljudske grupe, menja proces ažuriranja pozicija tragača i fazu učenja između podpopulacija, dok je faza inicijalizacije ostala nepromenjena. U narednom tekstu su detaljno opisane uvedene modifikacije.

3.1.1. Modifikacija ažuriranja pozicija tragača

Proces pretrage HSO algoritma traženja delimo u dve faze. Prva faza se odnosi na prvih 55% iteracija ovog algoritma gde je korišćena formula pretrage osnovnog ABC algoritma [54, 55]. Procenat iteracija koji je korišćen u dvema fazama pretrage HSO algoritma utvrđen je empirijski.

Pretraživanje tokom prve faze HSO algoritma je dato pomoću jednačine:

$$v_{ij} = x_{ij} + \varphi_{ij} \cdot (x_{ij} - x_{kj}) \tag{3.1}$$

gde x_{ij} označava j-ti parametar rešenja x_i, j je slučajni indeks, φ_{ij} je slučajni broj uniformne raspodele iz opsega [-1, 1], x_j predstavlja drugo slučajno rešenje iz populacije, $i = 1, 2, ..., SP$ i $j = 1, 2, ..., D$ (SP je broj rešenja u populaciji, a D je dimenzija problema).

Nakon kreiranja i evaluacije svakog potencijalnog rešenja, novo rešenje se poredi sa starim i primenjuje se mehanizam pohlepne selekcije. Ako novo rešenje ima manju vrednost funkcije cilja od tekućeg rešenja, tekuće rešenje se zamenjuje novim, u suprotnom se postojeće rešenje zadržava.

Druga faza pretrage se odnosi na preostalih 45% iteracija. U svakoj od ovih iteracija HSO algoritam bira između dve formule pretrage, modifikovane jednačine pretrage koja se koristi u SOA i modifikovane jednačine koja je korišćena u ABC algoritmu za rešavanje optimizacionih problema sa ograničenjima.

Razlika između jednačine pretrage koja se koristi u HSO algoritmu i jednačine 1.24 SOA je u računanju vektora δ_i, tj. menja se jednačina 1.21. U radu [22] je zaključeno da je vektor δ osetljiv parametar i da predložena jednačina 1.21 za njegovo računanje nije pogodna za optimizaciju multimodalnih funkcija. U cilju prevazilaženja ovog nedostatka za računanje ovog vektora predložena je jednačina [22]:

$$\delta'_{ij} = w \cdot |x_{best_j} - x_{rand_j}|, \ j = 1, 2, ..., D \tag{3.2}$$

gde je x_{best} pozicija najboljeg tragača u podpopulaciji kojoj pripada i-ti tragač, x_{rand} je vektor čije su vrednosti pozicije slučajno izabranog tragača iz iste podpopulacije kojoj tragač i pripada.

U SOA parametar težina inercije w se u toku jednog izvršavanja linearno smanjuje od vrednosti 0.9 do vrednosti 0.1 [22]. Na ovaj način raznovrsnost populacije se smanjuje kroz iteracije, tj. sposobnost eksploatacije SOA se povećava. U HSO algoritmu, u cilju održavanja veće raznovrsnosti populacije, parametar w se u toku ove faze algoritma linearno smanjuje od vrednosti 0.9 do vrednosti 0.75.

Može se zaključiti da je prva jednačina pretrage koja se koristi u ovoj fazi HSO algoritma (preostalih 45% iteracija) data pomoću:

$$v_{ij} = x_{ij} + \delta'_{ij} \cdot \sqrt{-\ln(\mu_{ij})} \cdot d_{ij},\ j = 1, 2, ..., D \tag{3.3}$$

U jednačini 3.3, v_{ij} je j-ti parametar potencijalnog novog i-tog rešenja populacije, x_{ij} je j-ti parametar tekućeg i-tog rešenja populacije, δ'_i je vektor koji se računa pomoću jednačine 3.2, μ_i je vektor stepena pripadnosti tragača koji se računa pomoću jednačine 1.22 i d_i je vektor dužine koraka koji se računa pomoću jednačine 1.18.

Druga jednačina pretrage koja je korišćena u ovoj fazi HSO algoritma je modifikovana jednačina pretrage koja je korišćena u ABC algoritmu za rešavanje optimizacionih problema sa ograničenjima. Ova jednačina je:

$$v_{ij} = \begin{cases} x_{ij} + rand_i \cdot (x_{ij} - x_{kj}) & , \text{ ako je } R_{ij} < 0.5 \\ x_{ij} & , \text{ inače} \end{cases} \tag{3.4}$$

U jednačini 3.4, v_{ij} je j-ti parametar potencijalnog novog i-tog rešenja populacije, x_{ij} je j-ti parametar tekućeg i-tog rešenja populacije, k je slučajno izabrani indeks iz podpopulacije kojoj pripada i -ti tragač i koji mora da bude različit od indeksa i, $rand_i$ je slučajni broj iz opsega [-1,1], R_{ij} je slučajni broj iz opsega (0,1] i $j = 1, 2, ...D$.

Razlika između jednačine pretrage koja se koristi u ABC algoritmu za rešavanje optimizacionih problema sa ograničenjima i jednačine 3.4 jeste što je vrednost $rand_i$ konstantna za svako $j = 1, 2, ...D$. Ova jednačina je uspešno korišćena za unapređenje performansi ABC algoritma za rešavanje inženjerskih problema [12]. U ovoj fazi pretrage nije korišćen mehanizam pohlepne selekcije, već se staro rešenje zamenjuje novim, nezavisno od vrednosti ciljne funkcije. Na ovaj način je povećana raznovrsnost populacije u ovoj fazi HSO algoritma.

U cilju određivanja jednačine pretrage koja se koristi u proizvoljnoj iteraciji druge faze pretrage, HSO algoritam uvodi novi kontrolni parametar koji nazivamo rata ponašanja (eng. behavior rate, BR). Ukoliko je slučajan broj iz opsega [0, 1] manji od vrednosti BR parametra koristi se jednačina 3.3, u suprotnom se koristi jednačina 3.4.

3.1.2. Modifikacija faze učenja između podpopulacija

U modifikovanoj fazi internog učenja između podpopulacija, jedinke koje imaju najveću vrednost funkcije cilja svake podpopulacije kombinuju se sa jedinkama koje imaju najmanju vrednost funkcije cilja iz drugih podpopulacija.

Neka vrednost parametra SN (eng. subpopulation number) označava ukupan broj podpopulacija HSO algoritma i neka vrednost parametra NSR (eng. number of seekers for replacement) označava broj tragača jedne podpopulacije koji imaju najveću vrednost ciljne funkcije. Pretpostavimo da su u svakoj podpopulaciji svi tragači sortirani u opadajućem redosledu prema vrednostima funkcije cilja.

Neka je l redni broj proizvoljne podpopulacije, $l \in \{1, 2, ..., SN\}$. U modifikovanoj fazi učenja između podpopulacija se NSR tragača podpopulacije l koji imaju najveću vrednost ciljne funkcije kombinuje redom sa tragačima koji imaju najmanju vrednost ciljne funkcije iz narednih NSR podpopulacija. Redni brojevi ovih podpopulacija su $(l + k)\ mod\ SN$, gde je $k = 1, 2, ..., NSR$.

Nova rešenja svake podpopulacije l, $l = 1, 2, ..., SN$ se kreiraju pomoću uniformnog operatora ukrštanja koji je dat pomoću jednačine:

$$v_{i_{worst,lj}} = \begin{cases} x_{i_{best,kj}} & , \text{ ako je } R_j < 0.5 \\ x_{i_{worst,lj}} & , \text{ inače} \end{cases} \tag{3.5}$$

U jednačini 3.5, R_j je slučajni realni broj iz opsega [0, 1], $x_{i_{worst,l}}$ označava poziciju i-tog najlošijeg rešenja l-te podpopulacije, $x_{i_{best,k}}$ označava poziciju najboljeg rešenja k-te podpopulacije, $i = 1, 2, ..., NSR$ i $k = (l + 1)\ mod\ SN, (l + 2)\ mod\ SN, ..., (l + NSR)\ mod\ SN$.

Nakon kreiranja NSR novih tragača svake podpopulacije, stare jedinke ovih podpopulacija se zamenjuju novim rešenjima. Treba napomenuti da se ova modifikovana faza učenja između podpopulacija primenjuje samo tokom poslednjih 45% iteracija HSO algoritma. Slična modifikacija faze učenja između podpopulacija je uspešno korišćena u radu [97].

Algorithm 7 Pseudo-kod HSO algoritma

Inicijalizovati kontrolne parametre SP, MCN, SN, BR i NSR
Inicijalizovati populaciju rešenja na slučajan način u prostoru pretrage x_{ij}, $i = 1 \ldots, SP$, $j = 1 \ldots, D$:
Izvršiti evaluaciju svakog rešenja x_i, $i = 1 \ldots, SP$
$t = 1$
$w = 0.9$
repeat
  if $(t < 0.55 \cdot MCN)$ **then**
    for $i = 1$ to SP **do**
      Kreirati potencijalno rešenje v_i za svako postojeće rešenje x_i koristeći jednačinu 3.1
      Izvršiti kontrolu ograničenja koja se odnose na granične vrednosti optimizacionih parametara kreiranog rešenja v_i koristeći jednačinu 1.7 i izvršiti evaluaciju ovog rešenja
      Primeniti proces pohlepne selekcije
    end for
  else
    for $i = 1$ to SP **do**
      if $(rand(0, 1) < BR)$ **then**
        Kreirati novo rešenje v_i populacije koristeći jednačinu 3.3
        Izvršiti kontrolu ograničenja koja se odnose na granične vrednosti optimizacionih parametara kreiranog rešenja v_i koristeći jednačinu 1.7 i izvršiti evaluaciju ovog rešenja
      else
        Kreirati novo rešenje v_i koristeći jednačinu 3.4
        Izvršiti kontrolu ograničenja koja se odnose na granične vrednosti optimizacionih parametara kreiranog rešenja v_i koristeći jednačinu 1.7 i izvršiti evaluaciju ovog rešenja
      end if
    end for
    Primeniti modifikovanu fazu učenja između podpopulacija koristeći jednačinu 3.5
    Ažurirati vrednost parametra w
  end if
  $t = t + 1$
until $t = MCN$

Prilikom analize rada algoritma pretraživanja ljudske grupe primenjenog za

optimizaciju multimodalnih funkcija, primećeno je da faza učenja između podpopulacija može dovesti do zaglavljivanja algoritma u lokalnom minimumu. Opisana modifikacija ove faze dovodi do povećanja raznovrsnosti populacije HSO algoritma u odnosu na SOA. Na ovaj način povećana je eksploracija unutar ove faze, a samim tim i u celom algoritmu, što dovodi do smanjenja mogućnosti da se algoritam zaglavi u lokalnom minimumu.

3.1.3. Pseudo-kod HSO algoritma

Pored zajedničkih kontrolnih parametara svih populacionih meaheuristika, veličina populacije SP i maksimalni broj iteracija MCN, HSO algoritam ima tri specifična kontrolna parametra. To su broj podpopulacija SN, rata ponašanja BR i broj tragača podpopulacije koji će biti zamenjeni novim tragačima NSR.

Pseudo-kod predloženog algoritma HSO dat je Algoritmom 7.

3.2. Benčmark funkcije

Za testiranje performansi predloženog HSO algoritma korišćen je skup od dvadeset tri benčmark funkcije. Korišenje velikog skupa test funkcija je uobičajeno u cilju obezbeđavanja fer poređenja performansi različitih optimizacionih algoritama. U slučaju da je skup test funkcija jako mali, postojala bi mogućnost da algoritam može da optimizuje samo odabrani skup funkcija. U tom slučaju bi izvođenje generalnog zaključka bilo teško. Skup test funkcija koje su korišćene, date su u radu [113]. Ovaj skup funkcija je dovoljno veliki da obuhvata različite vrste problema, kao što su jednomodalne, multimodalne, separabilne, neseparabilne i multidimenzionalne funkcije.

Za funkciju kažemo da je multimodalna ukoliko ima više od jednog lokalnog optimuma. Algoritam koji ima siromašan proces eksploracije ne može efikasno da istraži ceo prostor pretrage, zbog čega dolazi do zaglavljivanja u lokalnom optimumu. Funkcije koje imaju ravne površine su teške za algoritme kao i multimodalne funkcije, zbog toga što takve funkcije ne daju algoritmu nikakve informacije za usmeravanje procesa pretrage ka optimumu.

Drugu grupu test problema čine separabilne i neseparabilne funkcije. Proizvoljna separabilna funkcija koja ima D varijabli se može izraziti kao zbir D funkcija od jedne varijable [54]. Neseparabilne funkcije se ne mogu zapisati u ovoj formi. Ovu grupu test funkcija odlikuje povezanost između varijabli i značajno su teže za optimizaciju od separabilnih funkcija.

Tabela 3.1: Benčmark funkcije korišćene u eksperimentima

F	Naziv funkcije	C	D	S	f_{min}
f_1	Sphere	US	30	[-5.12, 5.12]^D	0
f_2	Schwefel 2.22	UN	30	[-10, 10]^D	0
f_3	Schwefel 1.2	UN	30	[-100, 100]^D	0
f_4	Schwefel 2.21	UN	30	[-100, 100]^D	0
f_5	Generalized Rosenbrock	UN	30	[-30, 30]^D	0
f_6	Step	US	30	[-100, 100]^D	0
f_7	Quartic Function with Noise	US	30	[-1.28, 1.28]^D	0
f_8	Generalized Schwefel 2.26	MS	30	[-500, 500]^D	-12569.5
f_9	Generalized Rastrigin	MS	30	[-5.12, 5.12]^D	0
f_{10}	Ackley	MN	30	[-32, 32]^D	0
f_{11}	Generalized Griewank	MN	30	[-600, 600]^D	0
f_{12}	Penalized	MN	30	[-50, 50]^D	0
f_{13}	Penalized2	MN	30	[-50, 50]^D	0
f_{14}	Shekel Foxholes	MS	2	[-65.54, 65.54]^D	0.998
f_{15}	Kowalik	MN	4	[-5, 5]^D	3.075e-4
f_{16}	Six Hump Camel Back	MN	2	[-5, 5]^D	-1.0316
f_{17}	Branin	MN	2	[-5, 15]^D	0.398
f_{18}	Goldstein-Price	MN	2	[-2, 2]^D	3
f_{19}	Hertman3	MN	3	[0, 1]^D	-3.86
f_{20}	Hertman6	MN	6	[0, 1]^D	-3.32
f_{21}	Shekel5	MN	4	[0, 10]^D	-10.1532
f_{22}	Shekel7	MN	4	[0, 10]^D	-10.4029
f_{23}	Shekel10	MN	4	[0, 10]^D	-10.5364

Skup benčmark funkcija koje su korišćene za testiranje predloženog HSO algoritma su prikazane u Tabeli 3.1. Karakteristike svake od ovih funkcija su date ispod kolone K. Ako je funkcija multimodalna, oznaka M je korišćena da naglasi ovu karakteristiku, dok U znači da je funkcija jednomodalna. Takođe, u ovoj koloni slovo S ističe da je funkcija separabilna, dok slovo N naglašava da je funkcija neseparabilna. Dalje, u Tabeli 3.1, D označava dimenziju test problema, S označava opseg varijabli i funkcija f_{min} predstavlja vrednost globalnog optimuma.

Funkcije od f_1 do f_{13} su problemi velikih dimenzija. Funkcije f_1-f_7 su jednomodalne. Funkcije f_8-f_{13} su multimodalne funkcije kod kojih broj lokalnih minimuma raste eksponencijalno sa dimenzijom problema, što ih čini najtežom klasom problema za mnoge optimizacione algoritme. Funkcije f_{14}-f_{23} su funkcije malih dimenzija koje imaju samo nekoliko lokalnih minimuma. Kod jednomodalnih funkcije brzina konvergencije algoritma je važnija od konačnih rezultata zbog toga što postoje druge metode koje su specifično dizajnirane za optimizaciju jednomodalnih funkcija. Multimodalne funkcije se koriste da bi testirale sposobnost algoritma da izbegne lokalni optimum i da pronađe dobro rešenje blisko optimalnom. Zbog toga je u slučaju ovih problema konačno rešenje mnogo važnije od brzine konvergencije.

Iz Tabele 3.1 se može videti da u ovom skupu funkcija imamo 16 multimodalnih problema i 7 jednomodalnih, od kojih je 19 neseperabilnih, dok su četiri funkcije separabilne.

3.3. Eksperimentalni rezultati

Predloženi HSO algoritam implementiran je u programskom jeziku Java 6 na Eclipse platformi SDK 3.7.2 i izvršavan je na računaru Intel(R) Core(TM) 2 Duo CPU E8500 sa 4 GHz i 4 GB RAM. ABC algoritam je takođe implementiran i testiran na ovom skupu test funkcija u cilju upoređivanja rezultata.

Performanse ovde predloženog HSO algoritma upoređene su sa performansama SOA [21] i ABC algoritmom. U radu [21] rezultati SOA upoređeni su sa rezultatima algoritma diferencijalne evolucije [94] i tri varijante algoritama PSO, PSO sa težinom inercije (eng. particle swarm optimization with inertia weight, PSO-w) [98], PSO sa faktorom suženja (eng. PSO with constriction factor, PSO-cf) [20] i optimizacijom čestica rojeva obimnim učenjem (eng. comprehensive learning particle swarm optimizer, CLPSO) [80]. Imajući u vidu da u radu [21] SOA nije pokazao konstantno bolje performanse od ovih algoritama, HSO algoritam je upoređen i sa svakim od njih.

Tabela 3.2: Statistički rezultati dobijeni pomoću algoritama DE [94], PSO-w [98], PSO-cf [20], CLPSO [80], ABC, SOA [21] i HSO tokom 50 nezavisnih pokretanja za funkcije f_1-f_4

Funkcija	Algoritam	Najbolje rešenje	Prosečno rešenje	Standardna devijacija
f_1	DE	5.20e-14	3.74e-13	3.94e-13
(MCN=1500)	PSO-w	1.79e-15	1.66e-13	4.59e-13
	PSO-cf	4.50e-45	2.28e-41	4.54e-41
	CLPSO	3.22e-13	2.73e-12	1.68e-12
	SOA	1.94e-83	1.02e-76	6.51e-76
	ABC	3.04e-16	5.09e-16	4.82e-17
	HSO	7.64e-26	1.42e-24	3.33e-24
f_2	DE	6.17e-10	3.74e-09	2.20e-09
(MCN=2000)	PSO-w	5.36e-12	6.67e-11	7.98e-11
	PSO-cf	3.29e-29	1.60e-00	4.22e-00
	CLPSO	1.63e-09	3.82e-09	1.73e-09
	SOA	3.40e-65	4.22e-63	8.25e-63
	ABC	1.85e-15	2.57e-15	5.01e-16
	HSO	4.84e-18	1.39e-17	8.52e-18
f_3	DE	1.10e-11	1.85e-10	1.49e-10
(MCN=5000)	PSO-w	2.00e-02	2.40e-01	2.23e-01
	PSO-cf	3.01e-19	3.33e+02	1.78e+03
	CLPSO	3.37e-02	4.20e-01	3.62e-01
	SOA	5.85e-35	4.26e-25	2.15e-24
	ABC	2.65e-16	4.21e-16	8.04e-17
	HSO	5.12e-86	1.21e-83	5.19E-83
f_4	DE	6.83e-13	3.10e-02	8.70e-02
(MCN=5000)	PSO-w	1.18e-02	7.02e-02	4.66e-02
	PSO-cf	1.48e-16	7.13e-13	2.19e-12
	CLPSO	6.88e-04	2.05e-03	1.25e-03
	SOA	1.55e-53	1.02e-48	2.46e-48
	ABC	6.56e-02	1.51e-01	2.12e-01
	HSO	1.04e-04	1.77e-02	3.65e-02

U svim eksperimentima, za sve algoritme, uzeta je ista veličina populacije 100, a izvršavanje algoritama je ponavljano 50 puta. Maksimalni broj iteracija MCN za svaki testirani problem je isti kod svakog algoritma i nji-

hove vrednosti su za svaku testiranu funkciju date u radu [113]. Dobijeni rezultati prikazani su u Tabelama 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 i 3.8.

Za ABC algoritam vrednost kontrolnog parametra *limit* je $SN \cdot D \cdot 0.5$. Za predloženi HSO broj podpopulacija SP podešen je na 10, rata ponašanja BR je 0.5 i broj tragača podpopulacije koji će biti zamenjeni novim tragačima NSR je $0.3 \cdot SP/SN$.

Tabela 3.3: Statistički rezultati dobijeni pomoću algoritama DE [94], PSO-w [98], PSO-cf [20], CLPSO [80], SOA [21], ABC i HSO tokom 50 nezavisnih pokretanja za funkcije f_5-f_7

Funkcija	Algoritam	Najbolje rešenje	Prosečno rešenje	Standardna devijacija
f_5	DE	0	3.47e-31	2.45e-30
(MCN=20000)	PSO-w	1.05e-02	1.82e+03	1.27e+03
	PSO-cf	1.87e-12	7.32e+03	2.46e+03
	CLPSO	1.68e-01	3.63e+01	3.12e+01
	SOA	2.42e+01	2.54e+01	7.87e-01
	ABC	8.96e-05	3.06e-02	4.68e-02
	HSO	1.90e-06	5.70e-04	1.77e-03
f_6	DE	0	0	0
(MCN=1500)	PSO-w	0	0	0
	PSO-cf	0	0	0
	CLPSO	0	0	0
	SOA	0	0	0
	ABC	0	0	0
	HSO	0	0	0
f_7	DE	1.97e-03	4.66e-03	1.30e-03
(MCN=3000)	PSO-w	2.99e-03	6.28e-03	2.17e-03
	PSO-cf	9.86e-04	2.45e-03	1.38e-03
	CLPSO	1.03e-03	2.98e-03	9.72e-04
	SOA	2.66e-05	1.08e-04	6.44e-05
	ABC	1.52e-02	2.95e-02	7.35e-03
	HSO	3.29e-04	8.87e-04	4.46e-04

Tabele 3.2 i 3.3 prikazuju najbolje vrednosti, prosečne vrednosti i standardne devijacije dobijene tokom 50 pokretanja ovih algoritama za jednomodalne test funkcije f_1-f_7. Iz ovih Tabela se može se videti da SOA postiže najbolje rezultate za 4 od 7 funkcija (f_1, f_2, f_4 i f_7) u pogledu preciznosti

Tabela 3.4: Statistički rezultati dobijeni pomoću algoritama DE [94], PSO-w [98], PSO-cf [20], CLPSO [80], SOA [21], ABC i HSO tokom 50 nezavisnih pokretanja za funkcije f_8-f_{10}

Funkcija	Algoritam	Najbolje rešenje	Prosečno rešenje	Standardna devijacija
f_8	DE	-12096	-11234	455.5
(MCN=9000)	PSO-w	-10495	-9363.3	445.3
	PSO-cf	-10398	-9026.1	656.9
	CLPSO	-12569	-12271	177.8
	SOA	-11760	-10126	669.5
	ABC	-12569	-12569	4.82e-17
	HSO	-12569	-12569	3.33e-24
f_9	DE	9.95e-00	8.10e+01	3.23e+01
(MCN=5000)	PSO-w	7.96e-00	2.10e+01	8.01e-00
	PSO-cf	2.69e+01	6.17e+01	1.84e+01
	CLPSO	2.85e-10	1.34e-09	8.57e-10
	SOA	0	0	0
	ABC	0	0	0
	HSO	0	0	0
f_{10}	DE	5.79e-08	1.71e-07	7.66e-08
(MCN=1500)	PSO-w	1.39e-07	1.66e-06	2.66e-06
	PSO-cf	2.67e-15	5.59e-01	7.30e-01
	CLPSO	3.31e-06	6.81e-06	1.94e-06
	SOA	-4.44e-15	-4.44e-15	0
	ABC	7.44e-11	6.43e-10	4.73e-10
	HSO	8.36e-11	2.13e-11	4.45e-11

pretrage i robustnosti. U odnosu na SOA, HSO postiže bolje rezultate za funkcije f_3 i f_5, dok za funkciju f_6 oba algoritma imaju iste performanse. Iako HSO nije dao bolje rezultate od SOA za većinu testiranih jednomodalnih funkcija, njegove performanse su značajno bolje za problem f_5. Kada uporedimo performanse HSO algoritma sa performansama ABC algoritma, uočava se da HSO postiže bolje ili jednake rezultate za sve testirane jednomodalne funkcije.

Tabele 3.4 i 3.5 prikazuju uporedne statističke rezultate ovih algoritama za multimodalne test funkcije koje imaju mnogo lokalnih minimuma (f_8-f_{13}). U radu [21] je primećeno da za multimodalne funkcije rezultati SOA

Tabela 3.5: Statistički rezultati dobijeni pomoću algoritama DE [94], PSO-w [98], PSO-cf [20], CLPSO [80], SOA [21], ABC i HSO tokom 50 nezavisnih pokretanja za funkcije f_{11}-f_{13}

Funkcija	Algoritam	Najbolje rešenje	Prosečno rešenje	Standardna devijacija
f_{11}	DE	0	4.44e-04	1.77e-03
(MCN=2000)	PSO-w	0	1.59e-01	2.19e-02
	PSO-cf	0	1.11e-02	1.25e-02
	CLPSO	1.64e-14	2.96e-04	1.46e-03
	SOA	0	0	0
	ABC	0	8.22e-17	8.55e-17
	HSO	0	6.66e-18	3.45e-17
f_{12}	DE	3.40e-15	3.67e-14	4.07e-14
(MCN=1500)	PSO-w	8.85e-15	2.21e-00	5.52e-00
	PSO-cf	1.57e-32	1.66e+01	1.81e+01
	CLPSO	8.80e-12	4.80e-11	3.96e-11
	SOA	3.04e-04	1.28e-02	7.67e-03
	ABC	3.31e-16	5.15e-16	6.44e-17
	HSO	6.67e-25	2.11e-23	2.77e-23
f_{13}	DE	4.13e-14	2.91e-13	2.88e-13
(MCN=1500)	PSO-w	8.23e-07	5.72e+02	3.57e+02
	PSO-cf	1.35e-32	2.40e+02	2.40e+02
	CLPSO	1.18e-10	6.42e-10	4.46e-10
	SOA	2.77e-03	1.89e-01	1.30e-01
	ABC	4.48e-16	5.60e-13	2.09e-11
	HSO	1.47e-22	2.05e-20	2.74e-20

nisu obećavajući, jer je uočeno da za ovaj tip problema SOA često ostaje u lokalnom minimumu. Iz ovih Tabela može se videti da HSO algoritam poboljšava performanse SOA za funkcije f_8, f_{12} i f_{13}, ali da takođe zadržava dobre performanse za probleme f_{10} i f_{11}. Za problem f_9, SOA i HSO imaju iste performanse. Kada se uporedi HSO sa ABC algoritmom, može se zaključiti da HSO postiže bolje ili jednake rezultate za sve testirane multimodalne probleme sa mnogo lokalnih minimuma. Dodatno, iz Tabela 3.4 i 3.5 je jasno da su performanse HSO algoritma značajno bolje nego performanse DE algoritma i tri varijanate PSO algoritama.

Tabele 3.6, 3.7 i 3.8 prikazuju uporedne statističke rezultate ovih algori-

Tabela 3.6: Statistički rezultati dobijeni pomoću algoritama DE [94], PSO-w [98], PSO-cf [20], CLPSO [80], SOA [21], ABC i HSO tokom 50 nezavisnih pokretanja za funkcije f_{14}-f_{17}

Funkcija	Algoritam	Najbolje rešenje	Prosečno rešenje	Standardna devijacija
f_{14}	DE	0.998	0.998	2.88e-16
(MCN=100)	PSO-w	0.998	1.026	1.52e-01
	PSO-cf	0.998	0.998	8.69e-13
	CLPSO	0.998	0.998	5.63e-10
	SOA	0.998	1.199	5.30e-01
	ABC	0.998	0.998	1.05e-15
	HSO	0.998	0.998	2.68e-14
f_{15}	DE	3.0749e-04	4.7231e-02	3.55e-04
(MCN=4000)	PSO-w	3.0749e-04	2.0218e-03	5.47e-03
	PSO-cf	3.0749e-04	2.0225e-03	5.47e-03
	CLPSO	3.2847e-04	5.3715e-04	6.99e-05
	SOA	3.0749e-04	3.0749e-04	1.58e-09
	ABC	3.0824e-04	3.9940e-04	5.15e-05
	HSO	3.0749e-04	3.0749e-04	5.94e-19
f_{16}	DE	-1.0316	-1.0316	6.77e-13
(MCN=100)	PSO-w	-1.0316	-1.0316	8.80e-12
	PSO-cf	-1.0316	-1.0316	5.92e-12
	CLPSO	-1.0316	-1.0316	8.50e-14
	SOA	-1.0316	-1.0316	6.73e-06
	ABC	-1.0316	-1.0316	9.47e-15
	HSO	-1.0316	-1.0316	1.01e-07
f_{17}	DE	0.39789	0.39789	1.14e-08
(MCN=100)	PSO-w	0.39789	0.39789	2.33e-12
	PSO-cf	0.39789	0.39789	5.25e-12
	CLPSO	0.39789	0.39789	1.08e-13
	SOA	0.39789	0.39838	5.14e-04
	ABC	0.39789	0.39789	8.42e-08
	HSO	0.39789	0.39789	4.30e-06

tama za multimodalne test funkcije koje imaju nekoliko lokalnih minimuma (f_{14}-f_{23}). Iz ovih Tabela se uočava da HSO algoritam postiže bolje rezultate u odnosu na SOA za funkcije f_{14} i $f_{19} - f_{23}$. Za ostale testirane multi-

Tabela 3.7: Statistički rezultati dobijeni pomoću algoritama DE [94], PSO-w [98], PSO-cf [20], CLPSO [80], SOA [21], ABC i HSO tokom 50 nezavisnih pokretanja za funkcije f_{18}-f_{21}

Funkcija	Algoritam	Najbolje rešenje	Prosečno rešenje	Standardna devijacija
f_{18}	DE	3	3	3.31e-15
(MCN=100)	PSO-w	3	3	2.50e-11
	PSO-cf	3	3	2.05e-11
	CLPSO	3	3	5.54e-13
	SOA	3	3.0001	1.17e-04
	ABC	3	3	6.77e-07
	HSO	3	3	3.38e-11
f_{19}	DE	-3.8628	-3.8628	1.97e-15
(MCN=100)	PSO-w	-3.8628	-3.8628	2.66e-11
	PSO-cf	-3.8628	-3.8628	2.92e-12
	CLPSO	-3.8628	-3.8628	6.07e-12
	SOA	-3.8628	-3.8621	6.69e-04
	ABC	-3.8628	-3.8628	5.00e-11
	HSO	-3.8628	-3.8628	5.31e-11
f_{20}	DE	-3.322	-3.215	0.036
(MCN=200)	PSO-w	-3.322	-3.256	0.066
	PSO-cf	-3.322	-3.277	0.058
	CLPSO	-3.322	-3.274	0.059
	SOA	-3.321	-3.298	0.045
	ABC	-3.322	-3.322	2.24e-11
	HSO	-3.322	-3.322	1.45e-11
f_{21}	DE	-10.15	-10.15	4.67e-06
(MCN=100)	PSO-w	-6.57	-2.01	1.10e-00
	PSO-cf	-10.15	-6.23	3.25e-00
	CLPSO	-10.14	-9.57	4.28e-01
	SOA	-10.15	-9.67	4.96e-01
	ABC	-10.15	-10.15	6.92e-04
	HSO	-10.15	-10.15	1.02e-04

modalne funkcije sa malim brojem lokalnih minimuma, HSO algoritam u odnosu na SOA postiže iste prosečne vrednosti koje su jednake optimalnim vrednostima, ali sa boljim standardnim devijacijama. Kada poredimo rezul-

Tabela 3.8: Statistički rezultati dobijeni pomoću algoritama DE [94], PSO-w [98], PSO-cf [20], CLPSO [80], SOA [21], ABC i HSO tokom 50 nezavisnih pokretanja za funkcije f_{22} i f_{23}

Funkcija	Algoritam	Najbolje rešenje	Prosečno rešenje	Standardna devijacija
f_{22}	DE	-10.40	-10.40	2.07e-07
(MCN=100)	PSO-w	-4.61	-2.14	8.34e-01
	PSO-cf	-10.40	-6.47	3.56e-00
	CLPSO	-10.34	-9.40	1.12e-00
	SOA	-10.40	-9.79	4.48e-01
	ABC	-10.40	-10.40	6.92e-04
	HSO	-10.40	-10.40	2.07e-05
f_{23}	DE	-10.54	-10.54	3.21e-06
(MCN=100)	PSO-w	-6.63	-2.20	1.01e-00
	PSO-cf	-10.54	-8.11	3.47e-01
	CLPSO	-10.46	-9.47	1.25e-00
	SOA	-10.54	-9.72	4.72e-01
	ABC	-10.54	-10.53	3.14e-03
	HSO	-10.54	-10.54	7.26e-04

tate HSO i ABC algoritama za ovu grupu test funkcija, iz prikazanih Tabela možemo zaključiti da oba algoritma imaju slične performanse. Jedino u slučaju funkcija f_{15} i f_{23}, HSO algoritam ima nešto bolje performanse. U poređenju sa DE algoritmom i tri varijante PSO algoritama, HSO algoritam postiže bolje ili jednake rezultate za većinu multimodalnih funkcija sa nekoliko lokalnih minimuma.

Generalno, dobijeni rezultati pokazuju da HSO algoritam ima bolje rezultate u odnosu na SOA, ABC, DE i tri varijante PSO algoritama, u odnosu na kvalitet rešenja i robustnost, za većinu testiranih funkcija.

Glava 4

Poboljšani FA za strukturnu optimizaciju

Nedavno predstavljen algoritam svitaca postao je jedan od istaknutih populacionih algoritama zbog njegove efikasnosti pri rešavanju širokog spektra realnih optimizacionih problema. U radu [36] FA je proširen za rešavanje strukturnih optimizacionih problema dodavanjem metode za kontrolu ograničenja koja je bazirana na kaznenim funkcijama. Rezultati optimizacije pokazali su da je FA efikasniji od drugih metaheuristika, kao što su PSO, GA, DE i SA.

U ovom delu disertacije predloženo je poboljšanje FA za rešavanje problema strukturne optimizacije koji sadrže i kontinualne i diskretne promenljive. Uvedene su dve modifikacije osnovnog algoritma. Jedna se odnosi na postupak upravljanja ograničenjima, zasnovan na Debovim pravilima. Druga se odnosi na uvođenje šeme redukcije geometrijske progresije, sa ciljem da se unaprede performanse algoritma u prostoru pretrage sa ograničenjima. Predloženi algoritam je testiran za rešavanje četiri klasična problema strukturne optimizacije, koja se nalaze u literaturi. Dobijeni rezultati pokazuju da ovaj pristup može biti veoma prihvatljiv za date probleme i da u većini slučajeva njegovi rezultati prevazilaze mogućnosti originalnog algoritma.

Poboljšani algoritam svitaca (eng. enhanced firefly algorithm, E-FA) je izložen u Odeljku 4.1. U Odeljku 4.2 su opisana četiri benčmark problema. Analiza postignutih rezultata izvršena je u Odeljku 4.3.

4.1. Predloženi E-FA

U predloženom E-FA uvedene su dve modifikacije u odnosu na osnovni FA. Prva modifikacija se odnosi na korišćenje tri pravila dopustivosti, odnosno

Debovih pravila, u cilju pronalaženja dopustivog regiona prostora pretrage. Druga modifikacija koristi redukcionu šemu geometrijske progresije radi smanjenja vrednosti faktora skaliranja S_k. U daljem tekstu detaljno su opisane ove modifikacije.

4.1.1. Modifikacija vezana za Debova pravila

Originalni FA nije dizajniran za rešavanje optimizacionih problema koji imaju ograničenja tipa nejednakosti ili jednakosti. Da bi generisao dopustiva rešenja originalni FA koristi metod kaznenih funkcija [36]. Metod kaznene funkcije je jedan od najčešće korišćenih metoda za upravljanje ograničenjima, jer problem sa ograničenjima svodi na problem koji ima samo ograničenja vezana za granice optimizacionih parametara.

Imajući u vidu da kaznene funkcije zahtevaju fina podešavanja vrednosti kaznenih faktora, koji su veoma zavisni od konkretnog problema koji treba optimizovati, E-FA koristi Debova pravila u cilju vođenja pretrage ka dopustivim regionima.

U predloženom E-FA, princip selekcije rešenja zasnovan na Debovim pravilima koristi se dva puta tokom kreiranja novog rešenja. Prvo se ova pravila koriste umesto pohlepne selekcije, da se odredi koji je svitac sjajniji. Zatim se Debova pravila koriste svaki put posle izvršenja jednačine 1.9 u cilju određivanja da li će se rešenje ažurirati.

U radu [53] je primećeno da populacioni algoritmi koji koriste Debova pravila imaju manju raznovrsnost populacije, zato što dopustiva rešenja imaju prednost nad nedopustivim. Imajući u vidu da se u E-FA Debova pravila koriste i kada se donosi odluka da li će novokreirano rešenje biti prihvaćeno, u ovom algoritmu raznovrsnost populacije je značajno smanjena u poređenju sa originalnim FA. Sa druge strane, mehanizam selekcije ima veliki uticaj na sposobnost pretrage populacionog algoritma [103].

Sa gledišta eksploracije i eksploatacije, smanjenje raznovrsnosti populacije tokom procesa pretrage uvećava sposobnost eksploatacije algoritma [103]. Takođe je uočeno, posmatranjem toka konvergencije opštih optimizacionih algoritama, da jača eksploatacija povećava brzinu konvergencije algoritma [106]. Zbog toga se može zaključiti da ova modifikacija povećava eksploataciju FA i, posledično, njegovu brzinu konvergencije.

4.1.2. Modifikacija vezana za faktore skaliranja

Kod originalnog FA algoritma primećeno je da se kvalitet rešenja može poboljšati redukcijom parametra randomizacije α [105]. Takođe je utvrđeno da je povećanje brzine konvergencije ostvarivo umanjivanjem vrednosti parametra randomizacije kako se algoritam približava optimalnom rešenju. Iz tog razloga je u radu [36] predložena redukciona šema geometrijske progresije za smanjenje parametra randomizacije. Uzimajući u obzir prethodne zaključke, kod E-FA se pokazalo da se kvalitet rešenja i brzina konvergencije mogu dalje poboljšati redukcijom svakog parametra skaliranja S_k, $k = 1, 2, ..., D$, korišćenjem iste redukcione šeme geometrijske progresije koja se koristi za smanjenje parametra α i koja je opisana jednačinom:

$$S_k(t) = S_k(t-1) \cdot \theta^{\frac{1}{MCN}} \tag{4.1}$$

gde je MCN maksimalni broj iteracija, t je tekući broj iteracije i θ je parametar koji se izračunava pomoću jednačine 1.14.

4.1.3. Pseudo-kod predloženog E-FA

Pseudo-kod predloženog algoritma E-FA dat je Algoritmom 8. Važno je primetiti da i FA i E-FA koriste isti broj kontrolnih parametara, imajući u vidu da podešavanje kontrolnih parametara može biti teži zadatak od samog problema koji se optimizuje [103]. Takođe, u cilju rešavanja strukturnih optimizacionih problema sa varijablama različitih tipova, kontinualne vrednosti diskretnih promenljivih se zaokružuju na najbliže raspoložive diskretne vrednosti, nakon svake primene jednačine 1.9, kao i nakon faze inicijalizacije.

4.2. Benčmark funkcije

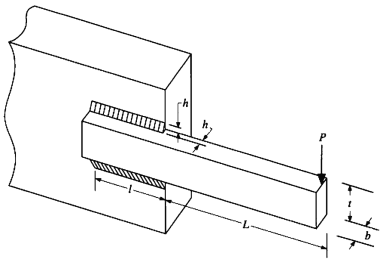
Predloženi E-FA primenjen je za rešavanje četiri strukturna optimizaciona problema koja su takođe korišćena u radu [36]. To su:

- Zavarena greda (eng. welded beam);
- Armirana betonska greda (eng. reinforced concrete beam);
- Dizajn zavojne opruge (eng. helical spring design) i
- Stepenasta konzolna greda (eng. stepped cantilever beam).

Algoritam 8 Pseudo-kod E-FA

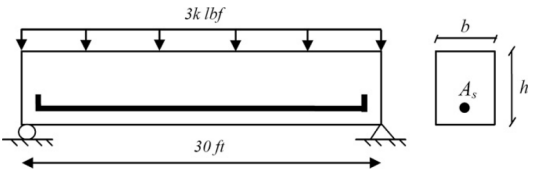
Inicijalizovati kontrolne parametre SP, MCN, α_0, γ i β_0
Inicijalizovati populaciju rešenja na slučajan način u prostoru pretrage x_{ij}, $i = 1 \ldots, SP$, $j = 1 \ldots, D$
Izvršiti evaluaciju svakog rešenja x_i, $i = 1 \ldots, SP$
Izračunati vrednosti S_k, $k = 1, 2, \ldots, D$ pomoću jednačine 1.12
$t = 1$
repeat
  for $i = 1$ to SP **do**
    for $j = 1$ to SP **do**
      if (x_j je izabrano na osnovu Debovih pravila kada se porede x_i i x_j) **then**
        for $k = 1$ to D **do**
          $g_k = x_{ik} + \beta \cdot (x_{ik} - x_{jk}) + \alpha \cdot S_k \cdot (rand_k - \frac{1}{2})$ {gde je β izračunato pomoću jednačine 1.10}
        end for
        Izvršiti kontrolu graničnih vrednosti parametara rešenja g koristeći jednačinu 1.7 i izvršiti evaluaciju ovog rešenja
        if (g je izabrano na osnovu Debovih pravila kada se porede x_i i g) **then**
          $x[i] = g$ {novo rešenje g je prihvaćeno umesto starog}
        end if
      end if
    end for
  end for
  for $k = 1$ to D **do**
    Izračunati vrednosti S_k pomoću jednačine 4.1
  end for
  Izračunati novu vrednost α pomoću jednačine 1.13
  Memorisati najbolje pronađeno rešenje
  $t = t + 1$
until $t = MCN$

Problem dizajna zavarene grede, prikazane na Slici 4.1, sastoji se u minimizaciji cene zavarivanja grede. Komponente koje predstavljaju ograničenja su: tangentni napon, napon savijanja grede, krivljenje opterećenja na gredi, deflekcija grede i bočna ograničenja. Četiri kontinualne promenljive su debljina vara h, dužina vara l, širina grede t i debljina grede b, gde su $0.1 \leq h, t \leq 2.0, \quad 0.1 \leq l, b \leq 10.0$. Optimalno rešenje je locirano na granicama dopustivog regiona, koji je veoma mali u odnocu na ceo prostor pretrage. Najbolje pronađeno rešenje za ovaj problem, zaokruženo na 7 decimala, iznosi 1.7248852 [63].



Slika 4.1: Skica problema zavarene grede

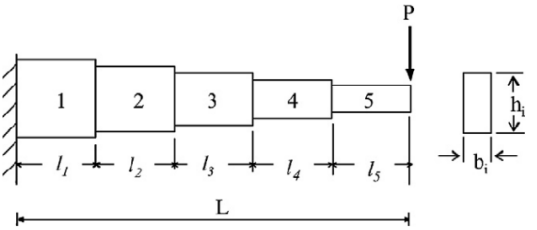
Problem dizajna armirane betonske grede, prikazane na Slici 4.2, se sastoji u minimizaciji cene. Greda je poduprta u dve tačke medjusobno udaljene 30 ft. Na gredu deluje živi teret od 2000 lbf i mrtvi teret od 1000 lbf (težina grede). Kod ovog problema promenljive su: površina poprečnog preseka armature A_s, širina betonske grede b i dubina betonske grede h. A_s je diskretna promenljiva koja podleže standardizaciji i mora se odabrati iz liste standardnih vrednosti [6]. Širina betonske grede b mora da bude celobrojna promenljiva iz skupa [28, 29, 30, 31, ..., 38, 39, 40]. Dubina betonske grede h je neprekidna promenljiva, $5 \leq h \leq 10$.



Slika 4.2: Skica problema armirane betonske grede

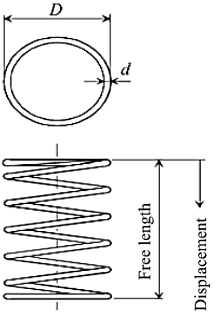
Problem dizajna stepenaste konzolne grede ima za cilj minimizaciju zapremine stepenaste konzolne grede izložene vertikalnoj sili i prikazan je na Slici 4.3. Visina i širina pravougaonog preseka svakog stepenastog

dela grede predstavljaju varijable ovog problema, tako da ukupno ima deset promenljivih. Širina b_1 i visina h_1 prvog dela (stepena) grede su celobrojne vrednosti. Širine b_2 i b_3 drugog i trećeg dela su diskretne vrednosti izabrane iz skupa [2.4, 2.6, 2.8, 3.1]. Visine h_2 i h_3 drugog i trećeg dela grede su diskretne vrednosti iz skupa [45.0, 50.0, 55.0, 60.0]. Ostale varijable ovog problema su neprekidne vrednosti.



Slika 4.3: Skica problema stepenaste konzolne grede

Problem dizajna zavojne opruge sastoji se u minimizaciji zapremine zavojne opruge od čelične žice. Ovaj problem, aktuelan u industriji izrade opruga, prikazan je na Slici 4.4. Problem uključuje 3 promenljive: broj kalemova opruge N, spoljni prečnik opruge D i prečnik žice opruge d. Broj kalemova opruge N je celobrojna promenljiva, spoljni prečnik D je neprekidna promenljiva, dok prečnik žice d može da uzima diskretne vrednosti prema dozvoljenim standardima za čelične žice [36].



Slika 4.4: Skica problema zavojne opruge

Matematički modeli ovih problema dati su u radu [36].

4.3. Eksperimentalni rezultati

Predloženi E-FA je implementiran u programskom jeziku Java na PC sa Intel(R) Core(TM) i7-3770K 4.2 GHz procesorom sa 16 GB RAM i Window 8x64 Pro operativnim sistemom. Dobijeni rezultati optimizacije E-FA upoređeni su sa odgovarajućim rezultatima koji su dobijeni originalnim FA [36].

4.3.1. Podešavanje parametara

Vrednosti parametara koje koristi originalni FA [36] su sledeće: veličina populacije $SP = 25$, vrednost parametra γ je $1/L$, gde je L tipična dužina varijabli, inicijalna vrednost atraktivnosti β_0 je 1, inicijalna vrednost parametra α je 0.5, dok je šemom opisanom jednačinom 1.13 redukovana vrednost parametra α sa 0.5 na 0.01. Za probleme zavarene grede i stepenaste konzolne grede korišćena je maksimalna vrednost broja ciklusa MCN= 2000. Za problem ojačane betonske grede MCN = 1000, dok je za problem zavojne opruge MCN = 3000. Statistički rezultati za FA dobijeni su iz 100 pokretanja ovog algoritma za svaki problem.

Vrednosti parametara korišćenih u predloženom E-FA, u svim eksperimentima, su sledeće: veličina populacije $SP = 25$, vrednost parametra γ je 1, inicijalna vrednost atraktivnosti β_0 je 1.5, a inicijalna vrednost parametra $\alpha = 0.9$. Za problem zavarene grede, maksimalni broj iteracija MCN je 200, dok je za probleme ojačane betonske grede i zavojne opruge MCN = 350. Za problem stepenaste konzolne grede vrednost MCN parametra je ograničena na 2000. Može se napomenuti da su performanse predloženog E-FA u slučaju izloženih problema osetljive na vrednost parametra β. Tako da je, iako je vrednost 1 za parametar β bila pogodna za većinu primena FA, u ovim eksperimentima korišćena nešto veća vrednost ovog parametra. Statistički rezultati za E-FA dobijeni su iz 100 pokretanja ovog algoritma za svaki problem.

4.3.2. Rezultati i diskusija

Upoređivanje i analiza rezultata zasnivaju se na dobijenim boljim najboljim rezultatima, boljim prosečnim vrednostima i vrednostima standardne devijacije, kao i manjem broju potrebnih evaluacija.

Najbolji rezultati demonstriraju mogućnost algoritma da dostigne optimalni rezultat, dok prosečna vrednost i standardna devijacija ukazuju na

Tabela 4.1: Uporedni rezultati dobijeni pomoću FA [36] i E-FA tokom 100 pokretanja za četiri problema strukturne optimizacije (najbolji rezultati su istaknuti)

Problem	Stats	FA	E-FA
Zavarena	Najbolje rešenje	1.7312065	**1.7248523**
greda	Prosečno rešenje	1.8786560	**1.7248523**
	Najlošije rešenje	2.3455793	**1.7248523**
	St. dev.	0.2677989	**6.17E-9**
	Eval.	50000	**5000**
Armirana	Najbolje rešenje	359.2080	359.2080
betonska	Prosečno rešenje	460.706	**359.2080**
greda	Najlošije rešenje	669.150	**359.2080**
	St. dev.	80.73870	**1.89E-10**
	Eval.	25000	**8750**
Stepenasta	Najbolje rešenje	**63893.52578**	64578.194035
konzolna	Prosečno rešenje	**64144.75312**	64578.194024
greda	Najlošije rešenje	**64262.99420**	64578.194053
	St. dev.	175.91879	**5.76E-6**
	Eval.	50000	50000
Zavojna	Najbolje rešenje	2.658575665	**2.658559166**
opruga	Prosečno rešenje	4.3835958	**2.6585592**
	Najlošije rešenje	7.8162919	**2.6585592**
	St. dev.	4.6076313	**4.42E-10**
	Eval.	75000	**8750**

robustnost ili stabilnost algoritma. Ukupan broj evaluacija se može smatrati cenom izračunavanja ili brzinom konvergencije.

Za prodloženi poboljšani algoritam svitaca dobijeni rezultati su prikazani pomoću vrednosti promenljivih visoke preciznosti da bi se obezbedila korektna provera vrednosti funkcija cilja i ograničenja dobijenih u eksperimentima. U tabeli 4.1 dati su uporedni statistički rezultati dobijeni korišćenjem originalnog algoritma svitaca [36] i predloženog poboljšanog algoritma. Bolji rezultati su istaknuti (eng. bold). U slučaju da oba algoritma imaju iste rezultate, nijedan rezultat nije istaknut.

Za problem zavarene grede, E-FA u svakom izvršavanju postiže do sada najbolje poznate zaokružene vrednosti [63]. Sa druge strane, za ovaj problem originalni algoritam svitaca je ostvario značajno slabije najbolje rešenje, kao i slabiju prosečnu vrednost i standardnu devijaciju. Dalje, E-FA je ove rezultate postigao za deset puta manji broj evaluacija u odnosu na FA.

Tabela 4.2: Vrednosti parametara i ograničenja za najbolja rešenja dobijena pomoću FA [36] i E-FA za problem zavarene grede (NA znači da vrednosti nisu dostupne)

	FA	E-FA
x_1	0.2015	0.20572963705932407
x_2	3.562	3.4704887163500215
x_3	9.0414	9.036623941581052
x_4	0.2057	0.20572963977307454
$g_1(x)$	NA	-1.26E-5
$g_2(x)$	NA	-2.05E-4
$g_3(x)$	NA	-2.71E-9
$g_4(x)$	NA	-3.432984
$g_5(x)$	NA	-0.080730
$g_6(x)$	NA	-0.235540
$g_7(x)$	NA	-1.24E-05
$f(x)$	1.7312	1.7248523165044531

Tabela 4.3: Vrednosti parametara i ograničenja za najbolja rešenja dobijena pomoću FA [36] i E-FA za problem dizajna armirane betonske grede

	FA	E-FA
x_1	6.32	6.32
x_2	34.0	34.0
x_3	8.5000	8.500000000000028
$g_1(x)$	-0.2241	-0.2241
$g_2(x)$	0	-1.33E-14
$f(x)$	359.2080	359.20800000000054

Za probleme dizajna armirane betonske grede i zavojne opruge, E-FA je došao do istih ili nešto boljih najboljih rezultata optimizacije i značajno boljih srednjih vrednosti i vrednosti standardne devijacije, u odnosu na FA. U odnosu na brzinu konvergencije, posmatrajući broj evaluacija, E-FA postiže ove rezultate oko tri puta brže za problem armirane betonske grede i oko osam puta brže za problem zavojne opruge u odnosu na originalni FA.

Za problem stepenaste konzolne grede, poboljšani algoritam svitaca, pri svakom izvršavanju postiže isto ili slično dopustivo najbolje rešenje čija zaokružena vredost iznosi 64578.194. U ovom slučaju originalni algoritam svitaca postiže bolju najbolju i prosečnu vrednost, ali znatno slabiju vrednost standardne devijacije. Broj evaluacija korišćen kod oba algoritma za

Tabela 4.4: Vrednosti parametara i ograničenja za najbolja rešenja dobijena pomoću FA [36] i E-FA za problem dizajna stepenaste konzolne grede (NA znači da vrednosti nisu dostupne)

	FA	E-FA
x_1	3	3
x_2	60	60
x_3	3.1	3.1
x_4	55	55
x_5	2.6	2.6
x_6	50	50
x_7	2.205	2.2808874178752334
x_8	44.091	45.61774832356247
x_9	1.750	1.7497570126997908
x_{10}	34.995	34.99514024843488
$g_1(x)$	NA	-1.38E-05
$g_2(x)$	NA	-1359.050
$g_3(x)$	NA	-153.846
$g_4(x)$	NA	-1203.412
$g_5(x)$	NA	-111.111
$g_6(x)$	NA	-1.16E-10
$g_7(x)$	NA	0.0
$g_8(x)$	NA	-2.258
$g_9(x)$	NA	-0.769
$g_{10}(x)$	NA	-1.49E-08
$g_{11}(x)$	NA	-3.18E-09
$f(x)$	63893.52	64578.19402431244

ovaj problem je isti. Ipak treba napomenuti da u slučaju originalnog FA [36] rezultati nisu prikazani pomoću vrednosti promenljivih visoke preciznosti, koji bi omogućili tačna izračunavanja dobijenih vrednosti funkcije cilja i ograničenja. Takođe, u radu [36] za problem stepenaste konzolne grede nisu prikazane vrednosti ograničenja.

Tabele 4.2, 4.3, 4.5 and 4.4 prikazuju parametre i vrednosti ograničenja za najbolja rešenja postignuta korišćenjem FA i E-FA.

Sumarni rezultati pokazuju da predloženi poboljšani algoritam svitaca nalazi najbolja rešenja koja su jednaka ili sasvim približna do sada najboljim pronađenim i objavljenim rezultatima u literaturi, za sva četiri testirana problema. U odnosu na originalni algoritam svitaca, poboljšani algoritam svitaca daje bolje rezultate sa gledišta kvaliteta i robustnosti, kao i značajno bržu konvergenciju za većinu testiranih problema.

Tabela 4.5: Vrednosti parametara i ograničenja za najbolja rešenja dobijena pomoću FA [36] i E-FA za problem dizajna zavojne opruge

	FA	E-FA
x_1	0.283	0.283
x_2	1.223049	1.2230410099640818
x_3	9	9
$g_1(x)$	-1008.02	-1008.811
$g_2(x)$	-8.946	-8.946
$g_3(x)$	-0.083	-0.083
$g_4(x)$	-1.777	-1.494
$g_5(x)$	-1.322	-1.322
$g_6(x)$	-5.464	-5.464
$g_7(x)$	0	0
$g_8(x)$	0.0000	-8.41E-13
$f(x)$	2.658575665	2.6585591659701966

Kako je u radu [36] iznet zaključak da je algoritam svitaca efikasniji od drugih metaheuristika kao što su PSO, GA, SA i harmony search (HS), to se takođe odnosi i na predloženi poboljšani algoritam svitaca, imajući u vidu da je za rešavanje ovih problema E-FA efikasniji od FA.

Glava 5

Adaptacija CS i FA za segmentaciju slika

Segmentacija slika je tehnika deljenja slike na sastavne delove ili objekte, koji su vizuelno različiti sa gledišta nekog svojstva, recimo nivoa sive boje ili teksture. Cilj segmentacije je da pojednostavi predstavljanje slika pomoću skupa segmenata koji su znatno pogodniji za dalje analize. Neke od praktičnih primena segmentacije slika uključuju kompjuterski vid (eng. computer vision) [44], industrijsku kontrolu kvaliteta [71], medicinske slike [4, 29, 43] i dr. Kako je problem segmentacije digitalnih slika jedno od važnih područja istraživanja, mnogo je tehnika segmentacije predloženo u literaturi.

Segmentacija slika tehnikom trešholding (eng. thresholding) je široko korišćena tehnika u obradi slika. Iscrpna pretraga je najčešće neprihvatljiva i skupa u realnom vremenu, jer broj izračunavanja eksponencijalno raste sa brojem pragova (eng. thresholds). Za rešavanje ovako teških optimizacionih problema uspešno se koriste populacione metaheuristike.

U ovom delu prikazana je adaptacija algoritma pretrage kukavice i algoritma svitaca za rešavanje problema segmentacije slike tehnikom trešholding. Algoritam diferencijalne evolucije i algoritam baziran na rojevima čestica su takođe implementirani u cilju poređenja rezultata segmentacije. Kao funkcije cilja korišćene su Kapurova entropija i međuklasna varijansa. Ova četiri algoritma su testirana na standardnim benčmark slikama sa globalnim optimumima dobijenim iz iscrpne pretrage (do pet trešhold vrednosti). Dobijeni rezultati segmentacije pokazuju da i algoritam svitaca i pretraga kukavice postižu bolje rezultate segmentacije u odnosu na druge dve populacione metaheuristike.

Kratak prikaz metoda za određivanje trešhold vrednosti je izložen u

Odeljku 5.1, dok je formulacija ovog problema data u Odeljku 5.2. Adaptacija CS i FA za segmentaciju slika izložena je u Odeljku 5.3. Analiza postignutih rezultata izvršena je u Odeljku 5.4.

5.1. Metode za određivanje trešhold vrednosti

Segmentacija slika tehnikom trešholding je veoma korišćen pristup u procesiranju slika, zbog svojih karakteristika i jednostavnosti primene ove tehnike. Ovaj pristup pokušava da izdvoji sadržaj slike od pozadine, na osnovu raspodele nijansi sive boje ili teksture objekta slike. Ako je slika podeljena na dve klase, pozadina i objekat od interesa, onda se ovaj pristup naziva trešholding na dva nivoa (eng. bi-level thresholding). Deljenje slike na više od dve klase naziva se trešholding na više nivoa (eng. multilevel thresholding).

Računanje trešhold vrednosti može da se vrši parametarskim ili neparametarskim metodama [116]. Parametarske metode podrazumevaju da je distribucija nivoa sive boje za svaku klasu u skladu sa zadatom raspodelom. Kod ovih metoda nalaženje optimalne trešhold vrednosti se izvršava putem procene parametara distribucije korišćenjem datog histograma. Ovo se svodi na nelinearan problem koji je sa gledišta procesorskog vremena skup. Neparametarski metodi se zasnivaju na traženju trešhold vrednosti na osnovu optimizacije nekih funkcija kriterijuma. Pokazalo se da su ovo robustnije i pouzdanije metode od parametarskih.

Do sada je u literaturi predloženo više različih kriterijuma za određivanje trešhold vrednosti [88]. Kao najviše korišćene funkcije kriterijuma izdvajaju se Kapurova entropija i međuklasna varijansa [47]. Kriterijum zasnovan na Kapurovoj entropiji ima za cilj da maksimizuje zbir entropija za svaku od klasa [48]. Korišćenje maksimuma entropije kao kriterijuma optimizacije za rešavanje problema segmentacije slika tehnikom trešholding prvi je predložio Pun. Kasnije je Kapur pronašao neke nedostatke kod Punovih izvođenja i predložio ispravljenu i poboljšanu verziju. Postoje različite varijante ovog kriterijuma, kao što je informaciono-teorijski kriterijum zasnovan na Renijevoj entropiji [84] ili jedan objedinjeni metod predložen u radu [104].

Drugi važan kriterijum za računanje trešhold vrednosti je međuklasna varijansa koju je definisao Otsu [72]. Cilj ovog kriterijuma je maksimizovanje separabilnosti klasa, koje su merene zbirom međuklasnih varijansi. Otsuov metod daje zadovoljavajuće rezultate kada su brojevi piksela u svakoj klasi približni jedan drugom. Za trešholding na dva nivoa ovi kriterijumi olakšavaju efikasnost algoritama, dok kod optimizacije na više nivoa složenost

izračunavanje kod klasičnih algoritama raste eksponencijalno sa brojem nivoa.

Neefikasnost izračunavanja determinističkih metoda dovodi do upotrebe metaheuristika za traženje optimalnih threšhold vrednosti. Algoritam diferencijalne evolucije je primenjen za traženje trešhold vrednosti korišćenjem kriterijuma minimalne entropije ukrštanja [85]. Yin je predložio algoritam baziran na rojevima čestica za traženje trešhold vrednosti, korišćenjem, takođe, kriterijuma minimalne entropije ukrštanja [120]. Horng je primenio metaheuristiku pčelinjih kolonija za traženje trešhold vrednosti upotrebom kriterijuma maksimalne entropije [46]. Akai je predstavila uporednu studiju PSO i ABC algoritama za rešavanje problema segmentacije slika tehnikom trešholding koristeći Kapurov kriterijum i kriterijum međuklasne varijanse[4].

Sathya i Kayalvizhi istraživali su algoritam baziran na populaciji bakterija i njegovu modifikovanu verziju za rešavanje problema segmentacije slika tehnikom trešholding koristeći kriterijum maksimuma entropije i kriterijum međuklasne varijanse [86], [87]. Adaptacija i upoređivanje šest metaheuristika za rešavanje problema segmentacije slika tehnikom trešholding korišćenjem kriterijuma međuklasne varijanse izložena je u radu [42]. Eksperimentalni rezultati su pokazali da su algoritmi PSO i DE nadmašili rezultate optimizacije dobijene metaheuristikom baziranom na populaciji mrava, genetskim algoritmom, algoritmom simuliranog kaljenja i algoritmom tabu pretrage.

5.2. Formulacija problema trešholding

Segmentacija slika trešholding tehnikom uključuje trešholding na dva nivoa i trešholding na više nivoa. Cilj problema trešholding na dva nivoa je da odredi jednu threšhold vrednost koja deli piksele slike u dve grupe. Jedna grupa, G_0, uključuje one piksele kod kojih je nivo sive boje ispod threšhold vrednosti x, tj. tačke objekta. Druga grupa, G_1, uključuje ostatak, odnosno tačke pozadine. Trešholding na dva nivoa može se definisati kao:

$$\begin{aligned} G_0 &= \{(a,b) \in I \mid 0 \leq g(a,b) \leq x-1\} \\ G_1 &= \{(a,b) \in I \mid x \leq g(a,b) \leq L-1\} \end{aligned} \tag{5.1}$$

gde je $g(a,b)$ nivo sive boje tačke (a,b), x je threšhold vrednost, $L-1$ je ukupan broj nivoa sive boje.

Cilj trešholding tehnike na više nivoa je da odredi više od jedne trešhold vrednosti koje dele piksele na nekoliko grupa. Neka je L broj nivoa sive boje za datu sliku I i neka su ovi nivoi sive boje u opsegu $0, 1, , ..., L-1$. Tada problem trešholding na više nivoa možemo definisati kao:

$$\begin{aligned} G_0 &= \{(a,b) \in I \mid 0 \leq g(a,b) \leq x_1 - 1\} \\ G_1 &= \{(a,b) \in I \mid x_1 \leq g(a,b) \leq x_2 - 1\} \\ G_2 &= \{(a,b) \in I \mid x_2 \leq g(a,b) \leq x_3 - 1\}, ... \\ G_D &= \{(a,b) \in I \mid x_D \leq g(a,b) \leq L - 1\} \end{aligned} \tag{5.2}$$

gde je $g(a,b)$ nivo sive boje tačke (a,b), x_i $(i = 1, 2, ..., D)$ je vrednost i-te trešhold vrednosti, D je broj trešhold vrednosti, $L-1$ je ukupan broj nivoa sive boje.

Nalaženje optimalne trešhold vrednosti kod problema trešholding na dva nivoa nije skup postupak sa gledišta vremena računanja. Sa druge strane, selekcija više od nekoliko trešhold vrednosti zahteva visoku cenu izračunavanja, jer računska složenost determinističkih algoritama raste eksponencijalno sa brojem trešhold vrednosti. Metode nalaženja optimalnih trešhold vrednosti obično traže ove vrednosti po nekim funkcijama kriterijuma. Ove funkcije kriterijuma definišu se na osnovu slika i koriste trešhold vrednosti kao parametre. U sledećem tekstu su izložene dve funkcije kriterijuma koje su korišćene u ovoj disertaciji, kriterijum entropije (Kapurov metod) i međuklasna varijansa (Otsuov metod).

5.2.1. Kriterijum entropije

Kriterijum entropije predložio je Kapur u cilju rešavanja problema trešholding na dva nivoa [48]. Ovaj kriterijum tretira pozadinu slike i glavni objekat kao dva odvojena izvora signala. Smatra se da izvršeno optimalno razdvajanje pozadine od glavnog objekta slike kada zbir entropije ove dve klase dostigne maksimum.

Entropija diskretnog izvora signala se dobija iz raspodele verovatnoće, gde je P_i verovatnoća da je sistem u stanju i [73]. Kapurova ciljna funkcija se određuje iz histograma slike, označenog sa h_i, $i = 0, 1, ..., L-1$, gde h_i predstavlja broj piksela koji imaju nivo sive boje i. Normalizovani histogram za nivo i je:

$$P_i = \frac{h_i}{\sum_{j=0}^{L-1} h_j} \tag{5.3}$$

Cilj je odrediti maksimum funkcije:

$$f(t) = H_0 + H_1 \tag{5.4}$$

gde je

$$\begin{aligned} H_0 &= -\sum_{i=0}^{x-1} \frac{P_i}{w_0} \ln \frac{P_i}{w_0}, & w_0 &= \sum_{i=0}^{x-1} P_i \\ H_1 &= -\sum_{i=x}^{L-1} \frac{P_i}{w_1} \ln \frac{P_i}{w_1}, & w_1 &= \sum_{i=x}^{L-1} P_i \end{aligned} \tag{5.5}$$

Kapurov metod može da se proširi za rešavanje trešholding problema na više nivoa. U tom slučaju ovaj metod se konfiguriše kao D-dimenzioni optimizacioni problem za selekciju D trešhold vrednosti $[x_1, x_2, ..., x_D]$. Potrebno je maksimizovati funkciju cilja:

$$f(x_1, x_2, ..., x_D) = H_0 + H_1 + H_2 + ... + H_D \tag{5.6}$$

gde je

$$\begin{aligned} H_0 &= -\sum_{i=0}^{x_1-1} \frac{P_i}{w_0} \ln \frac{P_i}{w_0}, & w_0 &= \sum_{i=0}^{x_1-1} P_i \\ H_1 &= -\sum_{i=x_1}^{x_2-1} \frac{P_i}{w_1} \ln \frac{P_i}{w_1}, & w_1 &= \sum_{i=x_1}^{x_2-1} P_i \\ H_2 &= -\sum_{i=x_2}^{x_3-1} \frac{P_i}{w_2} \ln \frac{P_i}{w_2}, & w_2 &= \sum_{i=x_2}^{x_3-1} P_i, ... \\ H_D &= -\sum_{i=x_D}^{L-1} \frac{P_i}{w_D} \ln \frac{P_i}{w_D}, & w_D &= \sum_{i=x_D}^{L-1} P_i \end{aligned} \tag{5.7}$$

5.2.2. Kriterijum međuklasne varijanse

Metod međuklasne varijanse je predložio Otsu i to je jedna od najkorišćenijih metoda predloženih za thresholding na dva nivoa [72]. Ovaj algoritam podrazumeva da slika za koju treba izvršiti segmentaciju sadrži dve klase

piksela (glavni objekat i pozadinu). Zatim se računa optimalna treshold vrednost odvojeno za ove dve klase, tako da njihova zajednička, kombinovana međuklasna varijansa bude maksimalna.

Međuklasna varijansa se definiše kao zbir sigma funkcija svake klase. Cilj je naći maksimum funkcije:

$$f(t) = \sigma_0 + \sigma_1 \tag{5.8}$$

gde je

$$\sigma_0 = w_0(\mu_0 - \mu_t)^2, \qquad \mu_0 = \sum_{i=0}^{t-1} \frac{iP_i}{w_0}$$
$$\sigma_1 = w_1(\mu_1 - \mu_t)^2, \qquad \mu_1 = \sum_{i=t}^{L-1} \frac{iP_i}{w_1}$$

gde je $\mu_t = \sum_{i=0}^{L-1} iP_i$ prosečan intezitet originalne slike.

Otsu funkcija se jednostavno može proširiti na slučaj više nivoa. Na primer, za D klasa, cilj je maksimizovati funkciju:

$$f(x_1, x_2, ..., x_D) = \sigma_0 + \sigma_1 + ... + \sigma_D \tag{5.9}$$

gde je

$$\sigma_0 = w_0(\mu_0 - \mu_t)^2, \qquad \mu_0 = \sum_{i=0}^{x_1-1} \frac{iP_i}{w_0} \tag{5.10}$$
$$\sigma_1 = w_1(\mu_1 - \mu_t)^2, \qquad \mu_1 = \sum_{i=x_1}^{x_2-1} \frac{iP_i}{w_1}$$
$$\sigma_2 = w_2(\mu_2 - \mu_t)^2, \qquad \mu_2 = \sum_{i=x_2}^{x_3-1} \frac{iP_i}{w_2}, ...$$
$$\sigma_D = w_D(\mu_D - \mu_t)^2, \qquad \mu_D = \sum_{i=x_D}^{L-1} \frac{iP_i}{w_D}$$

5.3. Adaptacija CS i FA za trešholding na više nivoa

Metaheuristike CS i FA za rešavanje problema trešholding na više nivoa imaju za cilj da pronađu D-dimenzionalni vektor $[x_1, x_2, ..., x_D]$ koji maksimizuje funkciju cilja, koja je data jednačinom 5.6 u slučaju Kapurovog kriterijuma, odnosno jednačinom 5.9 u slučaju kriterijuma međuklasne varijanse. Funkcija podobnosti je jednaka funkciji cilja koja je opisana jednom od pomenute dve jednačine.

Algoritam pretrage kukavice primenjen za rešavanje problema trešholding na više nivoa se razlikuje od originalnog u primeni formule 1.17 na osnovu koje se izvršavaju izmene u populaciji na slučajan način. Ova formula je data pomoću jednačine:

$$v_{ij} = \begin{cases} x_{ij} + rand_1 \cdot (x_{permute1,j} - x_{permute2,j}) & , \text{ ako je } rand_2 > p_a \\ x_{ij} & , \text{ inače} \end{cases} \tag{5.11}$$

gde su $rand_1$ i $rand_2$ slučajni brojevi uniformne raspodele iz opsega $[0, 2]$ i $[0, 1]$, respektivno, $permute_1$ i $permute_2$ su dva različita niza dobijena pomoću funkcije permutacije koja je primenjena na skup $\{1, 2, ..., SP\}$ i $j = 1, 2, ..., D$.

U originalnom CS algoritmu, $rand_1$ je uniformni slučajni broj između 0 i 1. U adaptiranom CS algoritmu za rešavanje problema trešholding na više nivoa slučajan broj $rand_1$ je u opsegu [0,2]. Povećanjem opsega vrednosti promenljive $rand_1$ eksploracija CS algoritma se povećava na nivou faze u kojoj se menja populacija na slučajan način i posledično u celom algoritmu. Testiranjem je uočeno da ova mala modifikacija značajno poboljšava kvalitet rešenja i brzinu konvergencije CS algoritma.

Kod algoritma svitaca primenjenog za rešavanje problema trešholding na više nivoa, kao i u verziji FA predloženoj za rešavanje strukturnih problema optimizacije [36], pokazalo se da se kvalitet rešenja može poboljšati ukoliko se redukuje parametar randomizacije α pomoću jednačine 1.13.

Matricu potencijalnih rešenja CS i FA čine celobrojne vrednosti x_{ij}, $i = 1, 2, ..., SP$ i $j = 1, 2, ..., D$. Iz ovog razloga su vrednosti optimizacionih parametara ovih algoritama zaokruživane na najbliže cele brojeve nakon inicijalizacije, kao i nakon kreiranja novih rešenja.

Rad svakog algoritma je prekidan kada vrednost funkcije podobnosti najboljeg rešenja $f(t*)$ dostigne optimalnu vrednost f_{opt} dobijenu metodom iscrpne pretrage, tj. kada je $|f(t*) - f_{opt}| \leq \epsilon = 10^{-9}$, gde je ϵ maksimalno

dozvoljeno odstupanje. Ukoliko u nekom pokretanju algoritam ne pronađe rezultat dovoljno blizak optimalnom, rad algoritma se završava nakon maksimalnog izvršenog broja iteracija.

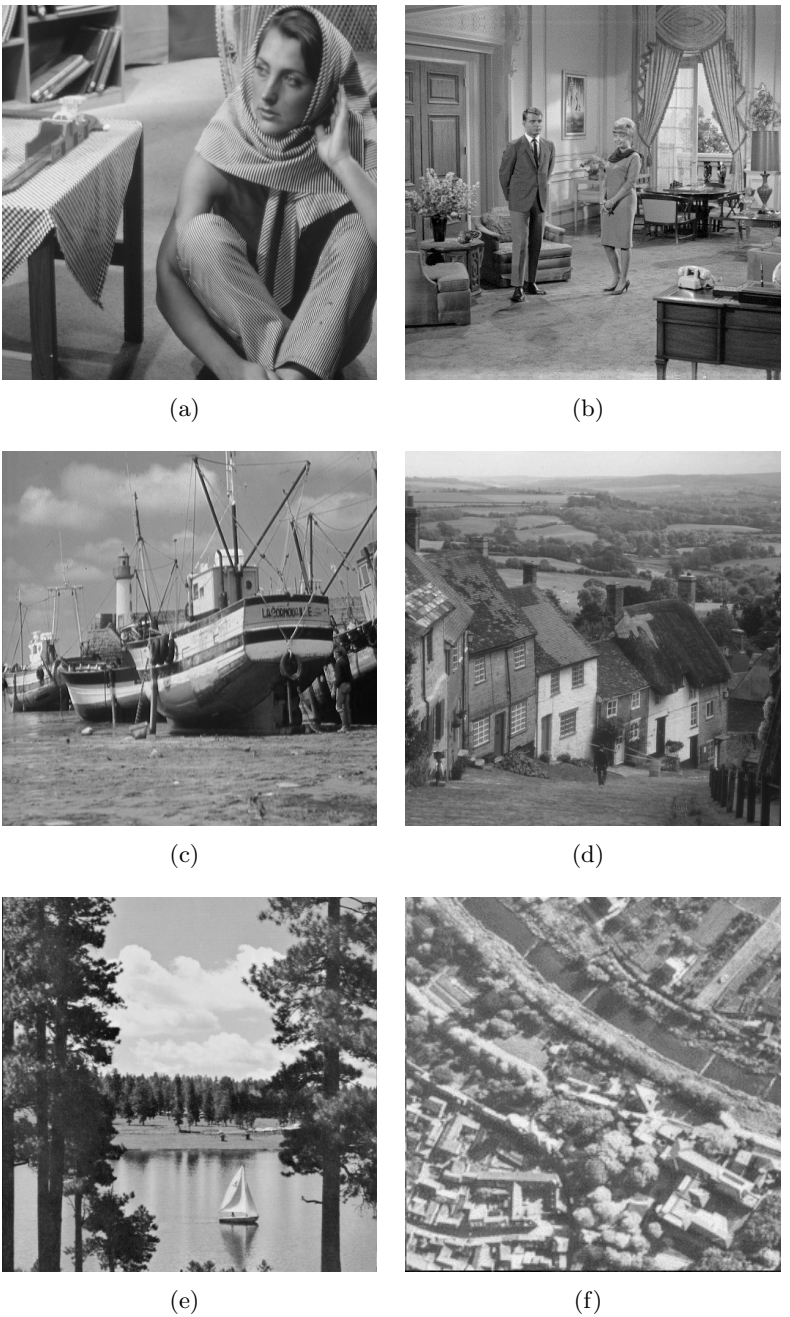
Pseudo-kodovi adaptiranih metaheuristika CS i FA dati su Algoritmima 9 i 10.

Algoritam 9 Pseudo-kod adaptiranog CS

Inicijalizovati kontrolne parametre SP, MCN i p_a
Inicijalizovati populaciju rešenja na slučajan način u prostoru pretrage x_{ij}, $i = 1 \ldots, SP$, $j = 1 \ldots, D$
Svaki optimizacioni parametar x_{ij} zaokružiti na najbliži ceo broj
Izvršiti evaluaciju svakog rešenja x_i, $i = 1 \ldots, SP$ pomoću jednačine 5.6 u slučaju Kapurovog kriterijuma ili pomoću jednačine 5.9 u slučaju kriterijuma međuklasne varijanse
$t = 1$
repeat
  for $i = 1$ to SP **do**
    for $j = 1$ to D **do**
      $v_{ij} = x_{ij} + 0.01 \cdot \left(\frac{r_1 \cdot \varphi}{r_2}\right)^{\frac{1}{\beta}} \cdot (x_{ij} - x_{best,j}) \cdot r_3$ {jednačina 1.16}
    end for
    Svaki optimizacioni parametar v_{ij} zaokružiti na najbliži ceo broj
    Izvršiti kontrolu graničnih vrednosti parametara rešenja v koristeći jednačinu 1.7 i izvršiti evaluaciju ovog rešenja
    Primeniti mehanizam "pohlepne" selekcije između rešenja x_i i v_i
  end for
  for $i = 1$ to SP **do**
    Primeniti jednačinu 5.11 za modifikovanje rešenja x_i
    Svaki optimizacioni parametar rešenja x_i zaokružiti na najbliži ceo broj
    Izvršiti kontrolu graničnih vrednosti parametara rešenja x_i koristeći jednačinu 1.7 i izvršiti evaluaciju ovog rešenja
  end for
  Memorisati najbolje pronađeno rešenje $f(x*)$
  $t = t + 1$
until ($t = MCN$ ili ($|f(t*) - f_{opt}| \leq \epsilon = 10^{-9}$))

Algoritam 10 Pseudo-kod adaptiranog FA

Inicijalizovati kontrolne parametre SP, MCN, α_0, γ i β_0
Inicijalizovati populaciju rešenja na slučajan način u prostoru pretrage x_{ij}, $i = 1 \ldots, SP$, $j = 1 \ldots, D$
Svaki optimizacioni parametar x_{ij} zaokružiti na najbliži ceo broj
Izvršiti evaluaciju svakog rešenja x_i, $i = 1 \ldots, SP$ pomoću jednačine 5.6 u slučaju Kapurovog kriterijuma, a pomoću jednačine 5.9 u slučaju kriterijuma međuklasne varijanse
Izračunati vrednosti S_k, $k = 1, 2, \ldots, D$ pomoću jednačine 1.12
$t = 1$
repeat
  for $i = 1$ to SP **do**
    for $j = 1$ to SP **do**
      if (x_j je izabrano na osnovu "pohlepne" selekcije x_i i x_j, tj. x_j ima veću vrednost funkcije cilja) **then**
        for $k = 1$ to D **do**
          $v_k = x_{ik} + \beta \cdot (x_{ik} - x_{jk}) + \alpha \cdot S_k \cdot (rand_k - \frac{1}{2})$ {gde je β izračunato pomoću jednačine 1.10}
          Svaki optimizacioni parametar v_k zaokružiti na najbliži ceo broj
        end for
        Izvršiti kontrolu graničnih vrednosti parametara rešenja v koristeći jednačinu 1.7 i izvršiti evaluaciju ovog rešenja
        $x_i = v$
      end if
    end for
  end for
  Izračunati novu vrednost α pomoću jednačine 1.13
  Memorisati najbolje pronađeno rešenje $f(x*)$
  $t = t + 1$
until ($t = MCN$ ili ($|f(t*) - f_{opt}| \leq \epsilon = 10^{-9}$))

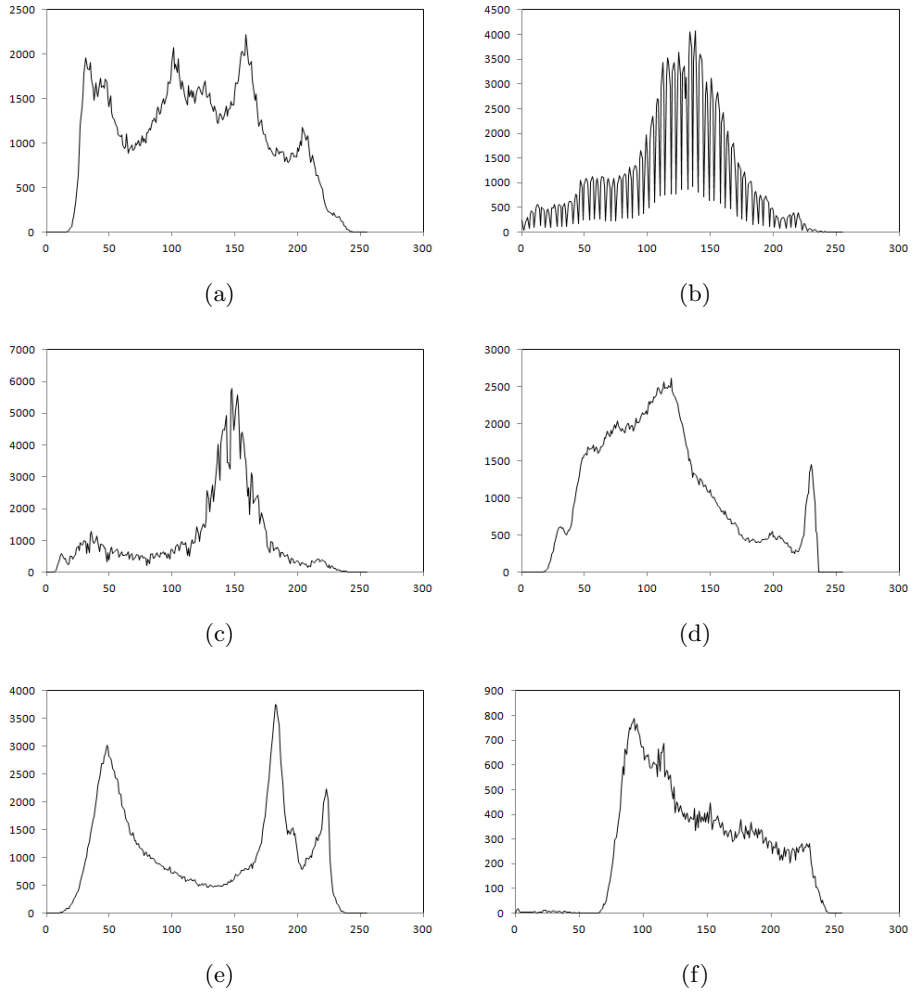


(a) (b)

(c) (d)

(e) (f)

Slika 5.1: Test slike: (a) Barbara, (b) Living room, (c) Boats, (d) Goldhill, (e) Lake, (f) Aerial.



Slika 5.2: Histogrami nijansi sive boje test slika: (a) Barbara, (b) Living room, (c) Boats, (d) Goldhill, (e) Lake, (f) Aerial.

5.4. Eksperimentalna studija

Metaheuristike CS i FA su adaptirane modifikacijama koje su opisane u prethodnom delu disertacije i implementirane u programskom jeziku Java. U programskom jeziku Java implementirani su i algoritmi za upoređivanje, DE i PSO. Sva četiri algoritma su pokretani na PC sa Intel(R) Core(TM) i7-3770K 4.2 GHz procesorom sa 16 GB RAM i Window 8x64 Pro operativnim sistemom.

Za ispitivanje je korišćen skup od šest standardnih test slika sa 256 nivoa sive boje. Ove test slike se često koriste u literaturi za procesiranje slika, sa nazvima Barbara, Living room, Boats, Goldhill, Lake i Aerial. Sve slike su veličine 512x512 piksela, osim slike Aerial koja je 256x256 piksela.

Test slike su prikazane na Slici 5.1. Njihovi histogrami su prikazani na Slici 5.2. Korišćene su dve različite funkcije kriterijuma, Kapurova entropija i međuklasna varijansa (Otsu funkcija). Za optimizaciju obe funkcije kriterijuma, izvršena je iscrpna pretraga do pet treshold vrednosti radi upoređivanja sa rezultatima generisanim metaheuristikama PSO, DE, CS i FA.

Broj treshold vrednosti k istraženih u ovim eksperimentima iznosi od 2 do 5. Kako metaheuristike imaju stohastičke karakteristike, svaki eksperiment je ponovljen 50 puta, za svaku od test slika i za svaku vrednost k. Optimalne threshold vrednosti, odgovarajuće vrednosti funkcije cilja, kao i vreme izvršavanja, potrebno da se iscrpnom pretragom dobiju ovi rezultati, koristeći Kapurov i Otsu kriterijum, date su u Tabeli 5.1 i Tabeli 5.2, respektivno.

5.4.1. Podešavanje parametara

Da bi se obezbedila fer komparacija, ista veličina populacije od 40 jedinki i maksimalan broj iteracija 2000 korišćeni su u svakom od četiri testirana algoritma. Pored ovih opštih kontrolnih parametara, svaki od testiranih algoritama ima svoje specifične kontrolne parametre, koji znatno utiču na njihov rad. Ovde je urađeno preliminarno testiranje algoritama, sa ciljem da se postigne dobra kombinacija vrednosti ovih parametara.

U slučaju PSO algoritma, vrednosti specifičnih kontrolnih parametara su: težina inercije $w = 0.5$, minimalna brzina $v_{min} = -5$, maksimalna brzina $v_{max} = 5$, faktor kognitivnog učenja $c_1 = 2$ i faktor socijalnog učenja $c_2 = 2$.

Za DE algoritam, korisćene su sledeće vrednosti specificnih kontrolnih parametara: faktor pojačanja razlike $F = 0.9$ i konstanta verovatnoće ukršta-

Tabela 5.1: Trešhold vrednosti, vrednosti funckije cilja i vreme obrade dobijeni pomoću iscrpne pretrage Kapurovim metodom

	k	Kapur		
		Trešhold vrednosti	Vrednost funkcije cilja	Vreme (ms)
Barbara	2	96, 168	12.668336540	28
	3	76, 127, 178	15.747087798	1812
	4	60, 99, 141, 185	18.556786861	122087
	5	58, 95, 133, 172, 210	21.245645311	5647294
Living room	2	94, 175	12.405985592	40
	3	47, 103, 175	15.552622213	1949
	4	47, 98, 149, 197	18.471055578	135259
	5	42, 85, 124, 162, 197	21.150302316	5875781
Boats	2	107, 176	12.574798244	32
	3	64, 119, 176	15.820902860	2063
	4	48, 88, 128, 181	18.655733570	126690
	5	48, 88, 128, 174, 202	21.401608305	5840989
Goldhill	2	90, 157	12.546393623	23
	3	78, 131, 177	15.607747002	2097
	4	65, 105, 147, 189	18.414213765	110650
	5	59, 95, 131, 165, 199	21.099138996	5064697
Lake	2	91, 163	12.520359742	31
	3	72, 119, 169	15.566286745	2094
	4	70, 111, 155, 194	18.365636309	119944
	5	64, 99, 133, 167, 199	21.024982760	5506378
Aerial	2	68, 159	12.538208248	30
	3	68, 130, 186	15.751881495	1880
	4	68, 117, 159, 200	18.615899102	118809
	5	68, 108, 141, 174, 207	21.210455499	5338382

Tabela 5.2: Trešhold vrednosti, vrednosti funckije cilja i vreme obrade dobijeni pomoću iscrpne pretrage Otsu metodom

	k	Otsu		
		Trešhold vrednosti	Vrednost funkcije cilja	Vreme (ms)
Barbara	2	82, 147	2608.610778507	12
	3	75, 127, 176	2785.163280467	786
	4	66, 106, 142, 182	2856.262131671	50854
	5	57, 88, 118, 148, 184	2890.976609405	2543860
Living room	2	87, 145	1627.909172752	13
	3	76, 123,163	1760.103018395	850
	4	56, 97, 132, 168	1828.864376614	59790
	5	49, 88, 120, 146, 178	1871.990616316	2731950
Boats	2	93, 155	1863.346730649	18
	3	73, 126, 167	1994.536306242	845
	4	65, 114, 147, 179	2059.866280428	54656
	5	51, 90, 126, 152, 183	2092.775965336	2654269
Goldhill	2	94, 161	2069.510202452	12
	3	83, 126, 179	2220.372641501	809
	4	69, 102, 138, 186	2295.380469158	51381
	5	63, 91, 117, 147, 191	2331.156597921	2573412
Lake	2	85, 154	3974.738214185	13
	3	78,140,194	4112.631097687	877
	4	67, 110, 158, 198	4180.886161109	60693
	5	57, 88, 127, 166, 200	4216.943583790	2740200
Aerial	2	125, 178	1808.171050536	11
	3	109, 147, 190	1905.410606582	801
	4	104, 134, 167, 202	1957.017965982	55015
	5	99, 123, 148, 175, 205	1980.656737348	2463969

Tabela 5.3: Upoređivanje prosečnih vrednosti i standardnih devijacija dobijenih pomoću metaheuristika PSO, DE, CS i FA baziranih na kriterijumu Kapurove entropije za slike Barbara, Living room, Boats i Goldhill za 50 pokretanja.

Alg.	k	Barbara		Living room	
		Prosečna vred.	St. dev.	Prosečna vred.	St. dev.
PSO	2	12.668336540	5.33E-15	12.405792709	1.64E-04
	3	15.747087798	1.42E-14	15.552015642	2.82E-03
	4	18.549612938	1.94E-02	18.467328310	6.64E-03
	5	21.241857967	6.71E-03	21.131564234	2.18E-02
DE	2	12.668336540	5.33E-15	12.405965639	7.89E-05
	3	15.747087798	1.42E-14	15.552578874	3.03E-04
	4	18.556749938	1.51E-04	18.470970822	3.16E-04
	5	21.245566656	2.34E-04	21.149062508	1.79E-03
CS	2	12.668336540	5.33E-15	12.405985592	5.33E-15
	3	15.747087798	1.42E-14	15.552622213	1.07E-14
	4	18.556786861	2.49E-14	18.471055578	2.49E-14
	5	21.245645311	1.42E-14	21.149400604	1.64E-03
FA	2	12.668336540	5.33E-15	12.405985592	5.33E-15
	3	15.747087798	1.42E-14	15.552622213	1.07E-14
	4	18.556786861	2.49E-14	18.471014902	2.85E-04
	5	21.245645311	1.42E-14	21.149483979	1.46E-03
Alg.	k	Boats		Goldhill	
		Prosečna vred.	St. dev.	Prosečna vred.	St. dev.
PSO	2	12.574798244	1.42E-14	12.546393623	7.11E-15
	3	15.820679619	8.84E-04	15.607747002	1.42E-14
	4	18.640100415	3.00E-02	18.414173744	2.07E-04
	5	21.392020144	4.12E-02	21.099092699	1.28E-04
DE	2	12.574798244	1.42E-14	12.546393623	7.11E-15
	3	15.820899807	1.42E-05	15.607743578	2.40E-05
	4	18.655660844	1.64E-04	18.414193378	8.19E-05
	5	21.401458219	3.21E-04	21.098959164	3.76E-04
CS	2	12.574798244	1.42E-14	12.546393623	7.11E-15
	3	15.820902860	8.88E-15	15.607747002	1.42E-14
	4	18.655733570	1.07E-14	18.414193014	7.04E-05
	5	21.401608305	7.11E-15	21.099125539	6.59E-05
FA	2	12.574798244	1.42E-14	12.546393623	7.11E-15
	3	15.820902860	8.88E-15	15.607747002	1.42E-14
	4	18.655723798	4.79E-05	18.414213765	2.13E-14
	5	21.401583877	7.33E-05	21.099138996	0.00E-00

Tabela 5.4: Upoređivanje prosečnih vrednosti i standardnih devijacija dobijenih pomoću metaheuristika PSO, DE, CS i FA baziranih na kriterijumu Kapurove entropije za slike Lake i Aerial za 50 pokretanja.

Alg.	k	Lake		Aerial	
		Prosečna vred.	St. dev.	Prosečna vred.	St. dev.
PSO	2	12.520359742	5.33E-15	12.538208248	1.78E-15
	3	15.566286745	1.24E-14	15.751881495	5.33E-15
	4	18.357505953	2.02E-02	18.615899102	1.78E-14
	5	21.015922726	4.40E-02	21.192396874	5.44E-02
DE	2	12.520359742	5.33E-15	12.538208248	1.78E-15
	3	15.566286745	1.24E-14	15.751881495	5.33E-15
	4	18.365579671	2.79E-04	18.615769177	6.37E-04
	5	21.024847591	3.20E-04	21.209823551	2.66E-02
CS	2	12.520359742	5.33E-15	12.538208248	1.78E-15
	3	15.566286745	1.24E-14	15.751881495	5.33E-15
	4	18.365636309	1.78E-14	18.615899102	1.78E-14
	5	21.024962923	5.95E-05	21.210455499	1.78E-15
FA	2	12.520359742	5.33E-15	12.538208248	1.78E-15
	3	15.566286745	1.24E-14	15.751881495	5.33E-15
	4	18.365636309	1.78E-14	18.615899102	1.78E-14
	5	21.024982760	0.00E-00	21.210455499	1.78E-15

nja $Cr = 0.9$. Ove vrednosti kontrolnih parametara za algoritme PSO i DE se koriste kod većine njihovih primena.

Kod ovde adaptiranog CS algoritma $p_a = 0.9$. Iako je vrednost ovog parametra 0.25 pogodna za većinu primena CS algoritma, ovde se pokazalo da je izvršavanje CS algoritma za problem threšholding osetljivo na vrednost ovog parametra i da su više vrednosti parametra p_a pogodnije.

Vrednosti parametara koje koristi adaptirani FA su sledeće: $\gamma = 1$, $\beta_0 = 1$, početna vrednost parametra $\alpha = 0.5$, tj. vrednost ovog parametra se iterativno smanjuje sa 0.5 na 0.01 pomoću redukcione šeme koja je data jednačinom 1.13.

5.4.2. Analiza kvaliteta rešenja

Za analizu kvaliteta rešenja, izračunate su prosečna vrednost i standardna devijacija kao rezultat 50 nezavisnih pokretanja svakog algoritma. Ovi rezultati su prikazani u Tabelama 5.3 i 5.4 za eksperimente zasnovane na Kapurovoj entropiji, odnosno u Tabelama 5.5 i 5.6 za eksperimente zasnovane na kriterijumu međuklasne varijanse. Dobijene prosečne vrednosti mogu se uporediti sa optimalnim vrednostima odgovarajućih funkcija cilja dobijenim metodom iscrpne pretrage koje su prikazane u Tabelama 5.1 i 5.2.

Iz Tabela 5.3 i 5.4 može se videti da u slučaju Kapurovog kriterijuma, metaheuristike CS i FA postižu bolje rezultate u odnosu na algoritme PSO i DE. Kod svih testiranh slika, za skoro svaki test slučaj, CS i FA dobijaju preciznije rezultate i robustniji su. Samo u nekim slučajevima, kada je broj trešhold vrednosti 2 ili 3, dobijeni su jednaki rezultati.

Kada se uporede performanse rezultata metaheuristika CS i FA za Kapurov kriterijum iz Tabela 5.3 i 5.4 se može videti da su njihovi rezultati slični sa gledišta pouzdanosti i robustnosti. Njihove prosečne vrednosti i standardne devijacije su jednake za svaku sliku kada je broj trešhold vrednosti manji od 4. Ovo važi i za slike Barbara, Aerial i za broj trešhold vrednosti $k = 4$ i $k = 5$. U ostalim slučajevima postoji mala razlika u njihovim performansama. Preciznije, u nekim slučajevima algoritam CS daje bolje rezultate (npr. slika Boats za broj trešhold vrednosti $k = 4$ i $k = 5$, ili Living room za broj trešhold vrednosti $k = 4$), dok u nekim slučajevima metaheuristika FA postiže nešto bolje rezultate (npr. kod slike Goldhill za broj trešhold vrednosti $k = 4$ i $k = 5$, Living room i Lake za broj trešhold vrednosti $k = 5$).

Rezultati dati u Tabelama 5.5 i 5.6, za kriterijum međuklasne varijanse, pokazuju da svi algoritmi imaju jednake performanse u pogledu pouzdanosti i robustnosti, za svaku sliku, kada je broj trešhold vrednosti $k = 2$. Za ovaj slučaj svi algoritmi postižu prosečne vrednosti jednake optimalnim, kao i standardnu devijaciju nula. Kada je broj trešhold vrednosti $k = 3$, metaheuristike PSO, CS i FA postižu optimalne vrednosti pri svakom pokretanju, za sve testirane slike. Inače, za ovaj slučaj testiranja, DE algoritam ima slabije rezultate za slike Barbara, Boats i Aerial u odnosu na ostale tri metaheuristike.

Kada je broj trešhold vrednosti veći od 3, iz rezultata prikazanih u Tabelama 5.5 i 5.6, vidi se da metaheuristike CS i FA ostvaruju bolje prosečne vrednosti i niže vrednosti standardne devijacije u odnosu na algoritme PSO i DE za sve testirane slike. Izuzetak je slika Goldhill za broj trešhold vred-

Tabela 5.5: Upoređivanje prosečnih vrednosti i standardnih devijacija dobijenih pomoću metaheuristika PSO, DE, CS i FA baziranih na kriterijumu međuklasne varijanse entropije za slike Barbara, Living room, Boats i Goldhill za 50 pokretanja.

Alg.	k	Barbara		Living room	
		Prosečna vred.	St. dev.	Prosečna vred.	St. dev.
PSO	2	2608.610778507	1.82E-12	1627.909172752	0.00E-00
	3	2785.163280467	2.27E-12	1760.103018395	2.27E-13
	4	2856.261511717	2.45E-03	1828.864376614	1.59E-12
	5	2890.975549258	5.05E-02	1871.984827146	2.29E-02
DE	2	2608.610778507	1.82E-12	1627.909172752	0.00E-00
	3	2785.162093432	8.31E-03	1760.103018395	2.27E-13
	4	2856.261305066	2.80E-03	1828.860328016	1.30E-02
	5	2890.971346990	2.05E-02	1871.976701063	2.34E-02
CS	2	2608.610778507	1.82E-12	1627.909172752	0.00E-00
	3	2785.163280467	2.27E-12	1760.103018395	2.27E-13
	4	2856.261098415	3.10E-03	1828.864376614	1.59E-12
	5	2890.976540127	4.85E-04	1871.990230213	2.70E-03
FA	2	2608.610778507	1.82E-12	1627.909172752	0.00E-00
	3	2785.163280467	2.27E-12	1760.103018395	2.27E-13
	4	2856.262131671	4.55E-13	1828.864376614	1.59E-12
	5	2890.976609405	3.64E-12	1871.990616316	0.00E-00
Alg.	k	Boats		Goldhill	
		Prosečna vred.	St. dev.	Prosečna vred.	St. dev.
PSO	2	1863.346730649	0.00E-00	2069.510202452	4.55E-13
	3	1994.536306242	1.59E-12	2220.372641501	1.36E-12
	4	2059.866220175	4.22E-04	2295.380095430	1.48E-03
	5	2092.771150715	8.36E-03	2331.156479206	3.56E-04
DE	2	1863.346730649	0.00E-00	2069.510202452	4.55E-13
	3	1994.535269293	7.26E-03	2220.372641501	1.36E-12
	4	2059.865271461	6.85E-03	2295.378073095	1.42E-02
	5	2092.766907541	2.71E-02	2331.145127974	2.55E-02
CS	2	1863.346730649	0.00E-00	2069.510202452	4.55E-13
	3	1994.536306242	1.59E-12	2220.372641501	1.36E-12
	4	2059.866280428	1.36E-12	2295.380469158	2.27E-12
	5	2092.775817560	1.03E-03	2331.155240485	4.76E-03
FA	2	1863.346730649	0.00E-00	2069.510202452	4.55E-13
	3	1994.536306242	1.59E-12	2220.372641501	1.36E-12
	4	2059.866280428	1.36E-12	2295.380469158	2.27E-12
	5	2092.773515829	3.57E-03	2331.156597921	2.27E-12

Tabela 5.6: Upoređivanje prosečnih vrednosti i standardnih devijacija dobijenih pomoću metaheuristika PSO, DE, CS i FA baziranih na kriterijumu međuklasne varijanse za slike Lake i Aerial za 50 pokretanja.

Alg.	k	Lake		Aerial	
		Prosečna vred.	St. dev.	Prosečna vred.	St. dev.
PSO	2	3974.738214185	3.64E-12	1808.171050536	2.27E-13
	3	4112.631097687	4.55E-12	1905.410606582	1.14E-12
	4	4180.883976390	7.41E-03	1955.085619462	7.65E+00
	5	4216.942888298	3.99E-03	1979.170306260	2.51E+00
DE	2	3974.738214185	3.64E-12	1808.171050536	2.27E-13
	3	4112.631097687	4.55E-12	1905.410561743	3.14E-04
	4	4180.884670497	5.62E-03	1957.014977950	1.18E-02
	5	4216.936401364	1.47E-02	1980.627553925	1.23E-01
CS	2	3974.738214185	3.64E-12	1808.171050536	2.27E-13
	3	4112.631097687	4.55E-12	1905.410606582	1.14E-12
	4	4180.886161109	0.00E-00	1957.017965982	0.00E-00
	5	4216.943583790	9.09E-13	1980.651043072	1.16E-02
FA	2	3974.738214185	3.64E-12	1808.171050536	2.27E-13
	3	4112.631097687	4.55E-12	1905.410606582	1.14E-12
	4	4180.886161109	0.00E-00	1957.017965982	0.00E-00
	5	4216.943583790	9.09E-13	1980.656737348	9.09E-13

nosti $k = 5$, gde algoritam PSO baziran na Otsu kriterijumu ima nešto bolje performanse od CS algoritma baziranog na Otsu kriterijumu.

Kada se uporede metaheuristike FA i CS bazirane na Otsu kriterijumu, može se videti da ova dva algoritma imaju jednake performanse za sve slike kada je broj treshold vrednosti $k = 4$. Metaheuristika FA postiže bolje rezultate za većinu testiranih slika kada je broj treshold vrednosti $k = 5$. Izuzetak je Slika Boats kada je broj treshold vrednosti $k = 5$, gde algoritam CS postiže nešto bolje rezultate od metaheuristike FA.

Prema prosečnim vrednostima i vrednostima standardnih devijacija datih u Tabelama 5.3 ,5.4, 5.5 i 5.6 može se zaključiti da predložene metaheuristike CS i FA, zasnovane na Kapurovom i na kriterijumu međuklasne varijanse, imaju znatno bolje performanse u poređenju sa algoritmima PSO i DE. Prosečne vrednosti metaheuristika CS i FA su vrlo blizu optimalnim vrednostima koje su dobijene iscrpnom metodom. Takođe, u većini test slučajeva,

Tabela 5.7: Prosečno procesorko vreme (u milisekundama) i prosečan broj iteracija dobijenih pomoću metaheuristika PSO, DE, CS i FA baziranih na kriterijumu Kapurove entropije za šest test slika tokom 50 pokretanja

Alg.	k	Barbara		Living room		Boats	
		Vreme (ms)	Iter.	Vreme (ms)	Iter.	Vreme (ms)	Iter.
PSO	2	2.18	9.22	102.82	1165.14	3.08	11.84
	3	3.30	14.28	21.96	218.56	16.23	136.58
	4	49.00	495.36	77.23	853.62	123.62	1367.86
	5	88.16	1050.8	153.53	1725.22	74.46	814.22
DE	2	3.55	14.52	14.24	134.26	3.37	16.9
	3	6.23	29.30	8.09	70.64	17.66	152.28
	4	24.65	586.48	33.50	322.98	46.99	478.76
	5	47.99	527.96	77.09	801.22	65.43	683.26
CS	2	71.04	194.54	53.30	129.84	53.84	135.58
	3	150.71	420.42	125.48	322.42	128.92	330.22
	4	189.31	518.58	222.48	570.6	170.28	436.02
	5	301.65	786.64	499.42	1303.8	247.15	631.36
FA	2	15.01	11.96	17.02	11.9	16.39	12.32
	3	34.07	29.82	37.24	29.7	36.24	29.24
	4	43.30	38.6	50.25	77.9	54.68	117.8
	5	50.15	44.04	104.74	515.06	76.22	241.94
Alg.	k	Goldhill		Lake		Aerial	
		Vreme (ms)	Iter.	Vreme (ms)	Iter.	Vreme (ms)	Iter.
PSO	2	2.54	8.84	2.41	8.86	2.62	10.7
	3	3.18	13.54	3.65	14.58	3.24	13.9
	4	10.22	97.86	26.99	295.76	4.03	19.76
	5	23.56	258.5	77.64	893.98	58.54	695.92
DE	2	2.13	16.44	5.03	15.76	3.88	16.96
	3	7.92	69.28	6.26	29.82	6.48	30.94
	4	17.06	163.98	22.16	165.04	14.08	122.68
	5	43.38	486.12	69.94	722.44	44.55	489.34
CS	2	82.29	209.48	70.48	183.36	56.32	150.88
	3	149.68	441.64	138.23	375.66	104.75	292.22
	4	278.51	828.32	220.44	604.58	174.84	482.26
	5	285.32	835.42	336.29	915.92	193.21	532.76
FA	2	13.66	10.08	15.92	12.32	13.86	11.0
	3	32.08	28.66	34.56	29.42	32.83	29.18
	4	42.24	38.6	43.84	37.66	43.18	38.96
	5	47.12	43.54	50.31	43.6	50.46	45.46

Tabela 5.8: Prosečno procesorko vreme (u milisekundama) i prosečan broj iteracija dobijenih pomoću metaheuristika PSO, DE, CS i FA baziranih na kriterijumu međuklasne varijanse za šest test slika tokom 50 pokretanja

Alg.	k	Barbara		Living room		Boats	
		Vreme (ms)	Iter.	Vreme (ms)	Iter.	Vreme (ms)	Iter.
PSO	2	0.84	9.4	0.90	9.6	0.94	8.8
	3	1.22	13.26	1.24	14.68	1.28	14.1
	4	4.84	136.66	1.60	19.18	3.12	56.56
	5	10.02	261.06	8.14	221.64	32.96	1012.3
DE	2	0.77	14.16	1.03	15.84	0.99	16.54
	3	3.08	68.4	2.26	30.4	2.79	69.46
	4	8.26	200.84	11.10	281.02	4.73	121.68
	5	27.36	798.72	24.28	682.8	32.93	953.74
CS	2	35.80	179.0	30.03	223.34	32.74	204.04
	3	51.66	370.88	56.67	371.50	53.74	378.40
	4	100.78	725.38	83.43	578.94	76.98	595.66
	5	114.63	802.80	122.78	836.32	102.84	677.04
FA	2	8.16	12.02	8.72	12.54	7.17	10.64
	3	15.73	28.62	12.67	28.54	16.10	28.7
	4	17.84	37.7	19.16	38.78	18.43	38.2
	5	21.27	43.9	20.15	43.78	43.51	669.52
Alg.	k	Goldhill		Lake		Aerial	
		Vreme (ms)	Iter.	Vreme (ms)	Iter.	Vreme (ms)	Iter.
PSO	2	0.94	8.98	0.91	10.12	0.81	9.04
	3	2.86	14.3	1.17	13.48	1.29	15.16
	4	5.16	136.32	6.55	175.58	5.18	138.44
	5	7.52	222.92	7.01	179.38	20.21	622.38
DE	2	0.82	15.24	0.75	14.72	0.82	15.8
	3	1.52	28.88	1.90	29.18	3.18	70.02
	4	7.80	200.44	7.43	201.24	6.99	162.28
	5	23.80	684.12	21.55	603.88	18.63	527.08
CS	2	27.88	195.30	38.04	267.66	32.45	254.10
	3	59.98	424.62	49.68	381.36	52.78	395.44
	4	70.92	531.74	85.30	646.44	83.84	593.52
	5	159.69	1115.44	109.90	746.36	209.36	1487.06
FA	2	8.61	12.68	7.28	11.66	7.94	12.18
	3	15.61	27.78	13.38	30.04	15.56	27.7
	4	17.12	37.66	18.49	38.86	18.05	38.58
	5	20.93	44.54	21.27	43.12	20.58	44.68

Tabela 5.8: Prosečno procesorko vreme (u milisekundama) i prosečan broj iteracija dobijenih pomoću metaheuristika PSO, DE, CS i FA baziranih na kriterijumu međuklasne varijanse za šest test slika tokom 50 pokretanja

Alg.	k	Barbara		Living room		Boats	
		Vreme (ms)	Iter.	Vreme (ms)	Iter.	Vreme (ms)	Iter.
PSO	2	0.84	9.4	0.90	9.6	0.94	8.8
	3	1.22	13.26	1.24	14.68	1.28	14.1
	4	4.84	136.66	1.60	19.18	3.12	56.56
	5	10.02	261.06	8.14	221.64	32.96	1012.3
DE	2	0.77	14.16	1.03	15.84	0.99	16.54
	3	3.08	68.4	2.26	30.4	2.79	69.46
	4	8.26	200.84	11.10	281.02	4.73	121.68
	5	27.36	798.72	24.28	682.8	32.93	953.74
CS	2	35.80	179.0	30.03	223.34	32.74	204.04
	3	51.66	370.88	56.67	371.50	53.74	378.40
	4	100.78	725.38	83.43	578.94	76.98	595.66
	5	114.63	802.80	122.78	836.32	102.84	677.04
FA	2	8.16	12.02	8.72	12.54	7.17	10.64
	3	15.73	28.62	12.67	28.54	16.10	28.7
	4	17.84	37.7	19.16	38.78	18.43	38.2
	5	21.27	43.9	20.15	43.78	43.51	669.52
Alg.	k	Goldhill		Lake		Aerial	
		Vreme (ms)	Iter.	Vreme (ms)	Iter.	Vreme (ms)	Iter.
PSO	2	0.94	8.98	0.91	10.12	0.81	9.04
	3	2.86	14.3	1.17	13.48	1.29	15.16
	4	5.16	136.32	6.55	175.58	5.18	138.44
	5	7.52	222.92	7.01	179.38	20.21	622.38
DE	2	0.82	15.24	0.75	14.72	0.82	15.8
	3	1.52	28.88	1.90	29.18	3.18	70.02
	4	7.80	200.44	7.43	201.24	6.99	162.28
	5	23.80	684.12	21.55	603.88	18.63	527.08
CS	2	27.88	195.30	38.04	267.66	32.45	254.10
	3	59.98	424.62	49.68	381.36	52.78	395.44
	4	70.92	531.74	85.30	646.44	83.84	593.52
	5	159.69	1115.44	109.90	746.36	209.36	1487.06
FA	2	8.61	12.68	7.28	11.66	7.94	12.18
	3	15.61	27.78	13.38	30.04	15.56	27.7
	4	17.12	37.66	18.49	38.86	18.05	38.58
	5	20.93	44.54	21.27	43.12	20.58	44.68

metaheuristike CS i FA ostvaruju veoma malu standardnu devijaciju. Ovo ukazuje da su predloženi algoritmi stabilni, što je značajna karakteristika za njihovu primenu u realnom vremenu.

Generalno se može zaključiti da metaheuristika FA postiže nešto bolje rezultate u slučaju obe funkcije kriterijuma, u odnosu na rezultate postignute algoritmom CS. Preciznije, za oba kriterijuma, za svaku test sliku i za svaku trešhold vrednost ($k = 2, 3, 4, 5$) postoji 48 testiranih slučajeva. Za njih 35, oba algoritma postižu jednake rezultate. Za ostalih 13 slučajeva, metaheuristika FA postiže bolje rezultate za 9 test slučajeva, 4 test slučaja za Kapurovu entropiju i 5 test slučajeva za međuklasnu varijansu.

5.4.3. Analiza vremena izračunavanja

Kako je vreme izvršavanja značajna odlika za aplikacije u realnom vremenu, analizirano je utrošeno vreme izračunavanja za testirane algoritme. U Tabeli 5.7 i u Tabeli 5.8 su date prosečne vrednosti broja iteracija i prosečno procesorsko vreme potrebno svakom algoritmu da zadovolji kriterijum zaustavljanja, u slučaju Kapurovog i u slučaju kriterijuma međuklasne varijanse, respektivno.

Vreme izračunavanja u slučaju iscrpne pretrage, za Kapurov kriterijum i kriterijum međuklasne varijanse raste eksponencijalno sa porastom broja trešhold vrednosti (Tabela 5.1 i Tabela 5.2). Iz Tabela 5.1 i 5.2 vidimo da je vreme izračunavanja neophodno za nalaženje optimalnih rezultata iscrpnom pretragom za kriterijum međuklasne varijanse aproksimativno dva puta kraće u odnosu na Kapurov kriterijum.

Ipak, u slučaju oba kriterijuma ono je praktično neprihvatljivo kada je broj trešhold vrednosti veći od 3. Sa druge strane, rezultati iz Tabela 5.7 i 5.8 pokazuju da vremena izračunavanja koja su potrebna metaheuristikama CS i FA raste sa povećanjem broja trešhold vrednosti. Međutim, sa povećanjem broja trešhold vrednosti, vremena izračunavanja koja su potrebna ovim algoritmima rastu linearno, a ne eksponencijalno kao u slučaju iscrpne pretrage.

Kada uporedimo vreme izračunavanja trešhold vrednosti koje je potrebno CS algoritmu, sa vremenima izračunavanja koja su potrebna ostalim testiranim metaheuristikama, može se videti da su u skoro svim slučajevima, metaheuristike PSO, DE i FA efikasnije nego CS algoritam, za obe funkcije kriterijuma. Takođe je evidentno da su za sve slike vremena izvršavanja potrebna CS i FA metaheuristikama baziranih na kriterijumu međuklasne varijanse, kraća od vremena izvršavanja CS i FA metaheuristika baziranih na Kapurovom kriterijumu. Inače, vremena izvršavanja koja su potrebna

metaheuristikama PSO, DE i FA se ne razlikuju značajno.

Iz Tabela 5.7 i 5.8 takođe se može videti da predložena metaheuristika FA konvergira za znatno manji broj iteracija u poređenju sa algoritmima PSO, DE i CS. Izuzetak su neki slučajevi kada je broj trešhold vrednosti mali ($k \leq 3$). U tom slučaju metaheuristike FA, PSO i DE konvergiraju jednakom brzinom, tj. za približan broj iteracija.

Može se zaključiti da su, iako je vreme izračunavanja koje je potrebno algoritmu CS duže u poređenju sa vremenom izračunavanja koje je potrebno metaheuristici FA, oba algoritma skalabilna. To znači da su oba algoritma pogodna, efikasna i praktična u pogledu vremenske složenosti za rešavanje problema velikih dimenzija.

Glava 6

Zaključak

U ovoj disertaciji su detaljno predstavljene poboljšane varijante nekih odabranih i najuticajnijih populacionih metaheuristika za rešavanje optimizacionih problema sa ograničenjima. Algoritam CB-ABC kao poboljšana varijanta metaheuristike pčelinjih kolonija i algoritam HSO kao poboljšana varijanta algoritma populacije ljudske grupe dovode do generalnog unapređenja performansi njihovih originalnih predstavnika. Za rešavanje problema strukturne optimizacije izložen je algoritam E-FA kao poboljšan algoritam svitaca, dok su za rešavanje problema segmentacije slika tehnikom trešholding predstavljene adaptacije algoritma svitaca i pretrage kukavice.

Opisan je princip rada i struktura svakog referentnog algoritma, izloženi su njihovi pseudo-kodovi radi potpunog uvida u povezanost njihovih faza i izložen je uticaj pojedinih komponenti na njihovo funkcionisanje. Za svaku novu, poboljšanu varijantu algoritma, opisane su modifikacije koje se uvode i objašnjeni su razlozi zbog kojih su uvedene, kao i poboljšanja koja se obezbeđuju na taj način. Objašnjeno je podešavanje parametara koji kontrolišu izvršavanje algoritma, metode primenjene u pojedinim fazama i prikazani su pseudo-kodovi svakog poboljšanog algoritma.

Za testiranje performansi poboljšane metaheuristike pčelinjih kolonija i hibridne metaheuristike bazirane na populaciji ljudske grupe korišćeni su standardni skupovi benčmark funkcija koje su namenjene za konkurentno rešavanje optimizacionih problema sa ograničenjima. Poboljšani algoritam svitaca je testiran za rešavanje problema strukturne optimizacije koji imaju promenljive kombinovanih tipova, kontinualne i diskretne, dok su adaptirane varijante algoritma svitaca i pretrage kukavice testirane na standardnim benčmark slikama.

Upoređivanje rezultata svakog poboljšanog algoritma rađeno je u odnosu

na rezultate najboljih do tada objavljenih, odgovarajućih metaheuristika koje su tretirale iste probleme. Radi objektivnog upoređenja posebna pažnja je posvećena vrednostima standardnih inicijalnih parametara ovih algoritama, veličini populacije i maksimalnom broju iteracija.

Dobijeni rezultati predloženih poboljšanih populacionih metaheuristika prikazani su paralelno sa rezultatima odgovarajućih referentnih algoritama. U disertaciji su za svaki poboljšani algoritam prikazane tabele uporednih rezultata. Osnovni podaci za upoređivanje su najbolji dobijeni rezultati optimizacije, najbolji mogući rezultati (ukoliko su poznati, realizovani metodom iscrpne pretrage ili na neki drugi način), prosečne vrednosti dobijenih rezultata, standardna devijacija i broj evaluacija.

Predloženi CB-ABC algoritam je testiran na velikom skupu od 24 standardnih benčmark problema i 4 standardna problema inženjerskog dizajna. Ovaj skup test funkcija sadrži različite tipove funkcija cilja, kao što su linearne, kvadratne i nelinearne, kao i različite tipove ograničenja. Dobijenim rezultatima CB-ABC algoritam je demonstrirao bolje performanse u odnosu na originalni ABC algoritam i prethodno razvijene poboljšane varijante ABC algoritma za rešavanje optimizacionih problema sa ograničenjima. Sem toga, predloženi CB-ABC algoritam je poređen i sa 11 najuspešnijih metaheuristika koje su poređene sa ABC algoritmom i njegovim varijantama. Upoređivanjem rezultata, direktno ili indirektno, putem tranzitivnosti je pokazano da CB-ABC algoritam postiže bolji kvalitet rezultata, da je robustniji i da ima bržu konvergenciju u odnosu na ovih 11 metaheuristika.

U cilju da se poboljša razmena dobrih informacija između rešenja i na taj način uveća eksploatacija ABC algoritma, dva modifikovana operatora pretrage se koriste u istraživačkoj i posmatračkoj fazi, kao i uniformni operator ukrštanja u istraživačkoj fazi. Korišćenjem dinamičkog smanjenja tolerancije kod ograničenja oblika jednakosti, CB-ABC algoritam poboljšava mogućnost za pronalaženje dopustivih rešenja. Takođe, CB-ABC algoritam koristi mehanizam kontrole ograničenja koja se odnose na granične vrednosti optimizacionih parametara koji poboljšavaju pretragu u graničnim delovima prostora pretrage.

HSO algoritam je testiran na velikom skupu test funkcija za rešavanje problema globalne optimizacije. Skup benčmark funkcija je kompletan i obuhvata različite vrste problema, kao što su jednomodalne, multimodalne, separabilne, neseparabilne i multidimenzionalne funkcije. Rezultati pokazuju da HSO algoritam radi bolje od SOA, ABC, DE algoritama i tri varijante PSO algoritama, u odnosu na kvalitet rešenja i robustnost, za većinu testiranih funkcija.

U ranoj fazi predloženog hibridnog algoritma koristi se formula pretrage ABC algoritma razvijenog za rešavanje problema globalne optimizacije. U kasnijim iteracijama algoritam kombinuje jednačine pretrage algoritma pretraživanja ljudske grupe i modifikovanu jednačinu pretrage algoritma ABC predloženu za rešavanje optimizacionih problema sa ograničenjima. U ovim iteracijama se koristi i faza učenja između podpopulacija, korišćenjem uniformnog operatora ukrštanja. Sposobnost eksploatacije HSO algoritma je naglašenija u prvim iteracijama jer se koristi i operator pohlepne selekcije u cilju zadržavanja boljih rešenja. Ova hibridizacija je uspešno premostila tendenciju SOA da prevremeno konvergira ka lokalnom optimumu za multimodalne funkcije, što ukazuje na mogućnost da se daljim modifikacijama realizuju još bolji rezultati.

Predloženi E-FA je testiran na skupu standardnih strukturnih optimizacionih problema koji imaju i kontinualne i diskretne promenljive. Ovaj algoritam uvodi dve modifikacije koje utiču na povećavanje sposobnosti eksploatacije algoritma svitaca. Prva modifikacija se odnosi na korišćenje operatora selekcije koji je baziran na Debovim pravilima, a druga na korišćenje šeme za dinamičku redukciju parametara koji kontrolišu pretragu.

Upoređivanje ostvarenih rezultata pokazalo je da E-FA za većinu testiranih problema nalazi slična ili bolja najbolja rešenja uz niže srednje vrednosti i značajno bržu konvergenciju.

Kako je algoritam svitaca jedna od istaknutih, u novije vreme predloženih populacionih metaheuristika, koja se već uspešno koristi za rešavanje velikog broja teških optimizacionih problema, rezultati dobijeni putem E-FA ohrabruju dalje primene ovog algoritma i na druge optimizacione probleme sa ograničenjima, kao i mogućnosti njegove adaptacije za rešavanje problema koji imaju više od jedne funkcije cilja.

Adaptacije algoritma svitaca i algoritma pretrage kukavice predložene su u ovoj disertaciji kako bi se istražila njihova primena za rešavanje problema trešholding na više nivoa. Kao funkcije cilja korišćene su Kapurova entropija i međuklasna varijansa. Radi upoređivanja rezultata adaptiranih CS i FA, realizovane su i na iste probleme primenjene i dve druge poznate metaheuristike za rešavanje trešholding problema na više nivoa, PSO i DE. Testiranja su izvršena na šest standardnih slika za koje su poznati optimumi dobijeni postupkom iscrpne pretrage.

Dobijeni rezultati pokazuju da kada je traženi broj trešhold vrednosti 2 ili 3, rezultati kvaliteta segmentacije slika su slični za sve ove algoritme. Međutim, za obe kriterijumske funkcije, kako broj traženih trešhold vrednosti raste, tako superiornost CS i FA postaje uočljivija u odnosu na algo-

ritme PSO i DE, sa gledišta pouzdanosti i robustnosti, sa malom prednošću adaptiranog FA. Analiza vremena izračunavanja pokazuje da FA konvergira za najmanji broj iteracija u poređenju sa ostalim primenjenim algoritmima. Iako je u većini slučajeva CS algoritmu potrebno nešto duže vreme izračunavanja nego algoritmima FA, PSO i DE, vreme izračunavanja trešhold vrednosti je prihvatljivo i raste linearno u odnosu na povećanje dimenzija problema, za razliku od postupaka iscrpne pretrage gde je ovaj rast eksponencijalan.

Prema rezultatima ovog istraživanja može se zaključiti da se obe metaheuristike, CS i FA, mogu efikasno koristiti za rešavanje problema trešholding na više nivoa, zbog njihove pouzdanosti i robustnosti. Rezultati segmentacija koji se njima dobijaju obećavaju i ohrabruju istraživanja za njihovu primenu na druge složene probleme obrade slika u realnom vremenu.

Svakom pojedinačnom poboljšanju populacionih metaheuristika prethodio je opsežan rad na istraživanju i sistematizaciji brojnih relevantnih studija objavljenih u naučnim časopisima. Uočeno je da zastupljenost ovih publikacija u celokupnoj naučnoj literaturi iz oblasti računarskih nauka ubrzano raste, što je posledica sve većeg interesovanja za istraživanje metaheuristika i njihove brojne primene.

Najvažniji naučni doprinosi koji su rezultat ove disertacije su:

- Detaljan prikaz nekih istaknutih, u novije vreme predloženih populacionih metaheuristika i objašnjenje uticaja pojedinih komponenti na njihov rad.
- Uvođenje boljih varijanti nekih od najsavremenijih i do sada najšire prihvaćenih populacionih metaheuristika do danas.
- Detaljno objašnjenje uvedenih poboljšanja sa gledišta uspostavljanja boljeg odnosa sposobnosti eksploatacije i eksploracije.
- Najbolji rezultati skoro svih poboljšanih populacionih metaheuristika izloženih u ovoj disertaciji su u većini testiranih slučajeva bolji ili jednaki od rezultata metaheuristika sa kojima su upoređivani, a neki od njih su bolji od najboljih do sada poznatih rezultata.
- Većina poboljšanih metaheuristika predstavljenih u ovoj disertaciji je uvela relativno malo ili nije uvela nijedan novi kontrolni parametar u odnosu na originalne metaheuristike.

- Sa gledišta teorije populacionih metaheuristika, rezultati ove disertacije doprinose stavu da se populacione metaheuristike mogu uspešno koristiti za rešavanje šireg skupa različitih teških optimizacionih problema, a da su pri tome nezavisne od prirode problema koje rešavaju.

- Svaka poboljšana populaciona metaheuristika izložena u ovoj disertaciji predstavlja doprinos razvoju teorije odgovarajuće populacione metaheuristike, kao i razvoju teorije populacionih metaheuristika u celini.

Rezultati svih ovde predstavljenih poboljšanih populacionih metaheuristika objavljeni su u prestižnim međunarodnim časopisima [14, 15, 16, 102]. Njihov uticaj je potvrđen znatnim brojem citata u međunarodnim naučnim publikacijama.

Literatura

[1] B. Akay and D. Karaboga, *Artifcial bee colony algorithm for large-scale problems and engineering design optimization*, Journal of Intelligent Manufacturing 23 (4) (2012), 1001–1014.

[2] B. Akay and D. Karaboga, *A modified Artificial Bee Colony algorithm for real-parameter optimization*, Information Sciences 192 (2012), 120–142.

[3] B. Akay and D. Karaboga, *Solving Integer Programming Problems by Using Artificial Bee Colony Algorithm*, in: R. Serra and R. Cucchiara (Eds.), AI*IA 2009: Emergent Perspectives in Artificial Intelligence 5883, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 355–364.

[4] B. Akay, *A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding*, Applied Soft Computing 13 (6) (2013), 3066-3091.

[5] B. Akay and D. Karaboga, *A survey on the applications of artificial bee colony in signal, image, and video processing*, Signal, Image and Video Processing 9 (4) (2015), 967–990.

[6] A. H. Amir and T. Hasegawa, *Nonlinear mixed-discrete structural optimization*, Journal of Structural Engineering 115 (3) (1989), 626-646.

[7] A. Baykasoglu, L. Ozbakir and P. Tapkan, *Artificial bee colony algorithm and its application to generalized assignment problem* ,In: Swarm intelligence focus on ant and particle swarm optimization, I-Tech Education and Publishing, Vienna, Austria, 2007, pp. 113–144.

[8] T. Bäck, F. Hoffmeister and H.-P. Schwefel, *A Survey of Evolution Strategies*, in: R. K. Belew, L. B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, California, 1991, pp. 2-9.

[9] H.G. Beyer and H.P. Schwefel, *Evolution strategies - A comprehensive introduction*, Natural computing 1 (1) (2002), 3-52.

[10] M. Birattari, *The problem of tuning metaheuristics: a machine learning perspective*, Studies in Computational Intelligence, Vol. 197, Springer-Verlag Berlin Heidelberg, 2009.

[11] D.O. Boyer, C.H. Martfnez and N.G. Pedrajas, *Crossover operator for evolutionary algorithms based on population features*, Journal of Artificial Intelligence Research 25 (2005), 1–48.

[12] I. Brajevic and M. Tuba, *An upgraded artificial bee colony algorithm (ABC) for constrained optimization problems*, Journal of Intelligent Manufacturing 24 (4) (2013), 729–740.

[13] I. Brajevic, M. Tuba and M. Subotic, *Performance of the improved artificial bee colony algorithm on standard engineering constrained problems*, International Journal of Mathematics and Computers in Simulation 5 (2) (2011), 135–143.

[14] I. Brajevic, *Crossover-based artificial bee colony algorithm for constrained optimization problems*, Neural Computing and Applications, ISSN 0941-0643, DOI: 10.1007/s00521-015-1826-y, 15 p., 2015.

[15] I. Brajevic and M. Tuba, *Cuckoo Search and Firefly Algorithm Applied to Multilevel Image Thresholding*, In: Cuckoo Search and Firefly Algorithm (X.S. Yang,eds.), Studies in Computational Intelligence, Volume 516, Springer International Publishing, 2014, pp. 115-139.

[16] I. Brajevic and J. Ignjatović, *An enhanced firefly algorithm for mixed variable structural optimization problems*, Facta Universitatis: Series Mathematics and Informatics, 2015, (accepted for publication).

[17] I. Brajevic, *Artificial bee colony algorithm for the capacitated vehicle routing problem*, Proceedings of the European Computing Conference, ISBN 978-960-474-294-7, Paris, 2011, pp. 239–244.

[18] M. Cavazzuti, *Optimization Methods*, From Theory to Design Scientific and Technological Aspects in Mechanics, Springer Berlin Heidelberg, 2013.

[19] S.P. Chatzis and S. Koukas, *Numerical optimization using synergetic swarms of foraging bacterial populations*, Expert Systems with Applications 38 (12) (2011), 15332-15343.

[20] M. Clerc and J. Kennedy, *The particle swarmexplosion, stability, and convergence in a multidimensional complex space*, IEEE Trans. on Evolutionary Computation 6 (1) (2002), 58-73.

[21] C. Dai, Y. Zhu and W. Chen, *Seeker Optimization Algorithm*, In: Computational Intelligence and Security, Lecture Notes in Computer Science, Volume 4456, 2007, pp. 167–176.

[22] C. Dai, W. Chen, Y. Song and Y. Zhu, *Seeker optimization algorithm: a novel stochastic search algorithm for global numerical optimization*, Journal of Systems Engineering and Electronics, 21 (2) (2010), 300–311.

[23] C. Dai, W. Chen, Y. Zhu, Z. Jiang and Z. You, *Seeker optimization algorithm for tuning the structure and parameters of neural networks*, Neurocomputing, 74 (6) (2011), 876–883.

[24] C. Dai, W. Chen and Y. Zhu, *Seeker optimization algorithm for digital IIR filter design*, IEEE Transactions on Industrial Electronics, 57 (5) (2010), 1710–1718.

[25] C. Dai, Z. Cheng, Q. Li, Z. Jiang and J. Jia, *Seeker optimization algorithm for global optimization: A case study on optimal modelling of proton exchange membrane fuel cell (PEMFC)*, International Journal of Electrical Power and Energy Systems, 33 (3) (2011), 369–376.

[26] C. Dai, W. Chen, L. Ran, Y. Zhang and Y. Du, *Human Group Optimizer with Local Search*, In: Advances in Swarm Intelligence, Lecture Notes in Computer Science, Volume 6728, 2011, pp. 310–320.

[27] K. Deb, *An Efficient Constraint-handling Method for Genetic Algorithms*, Computer Methods in Applied Mechanics and Engineering 186 (2–4) (2000), 311–338.

[28] M. K. Dhadwal, S. N. Jung and C. J. Kim, *Advanced particle swarm assisted genetic algorithm for constrained optimization problems*, Computational Optimization and Applications 58 (3) (2014), 781-806.

[29] A.R. Dominguez and A.K. Nandi, *Detection of masses in mammograms via statistically based enhancement, multilevel-thresholding segmentation, and region selection*, Computerized Medical Imaging and Graphics 32(4) (2008), 304-315.

[30] M. Dorigo and T. Stützle, *Ant Colony Optimization*, MIT Press, (2004).

[31] I. Fister Jr., M. Perc, S.M. Kamal and I. Fister, *A review of chaos-based firefly algorithms: Perspectives and research challenges* , Applied Mathematics and Computation **252** (2015), 155-165.

[32] W. F. Gao, S. Y. Liu and L. L. Huang *A novel artificial bee colony algorithm based on modified search equation and orthogonal learning*, IEEE Transactions on Cybernetics 43 (3) (2013), 1011-1024.

[33] W. F. Gao, S. Y. Liu and L. L. Huang, *A novel artificial bee colony algorithm with Powell's method* , Applied Soft Computing 39 (9) (2013), 3763-3775.

[34] W. F. Gao, S. Y. Liu and L. L. Huang, *Enhancing artificial bee colony algorithm using more information-based search equations* , Information Sciences 270 (20) (2014), 112-133.

[35] W. Gong, Z. Cai and D. Liang, *Engineering optimization by means of an improved constrained differential evolution* , Computer Methods in Applied Mechanics and Engineering 268 (2014), 884-904

[36] A. H. Gandomi, X-S. Yang and A. H.Alavi, *Mixed variable structural optimization using Firefly Algorithm* , Computers and Structures **89** (2011), 2325-2336.

[37] A. H. Gandomi, X-S. Yang and A. H. Alavi *Cuckoo search algorithm: a meta-heuristic approach to solve structural optimization problems*, Engineering with Computers 29 (1), Neural Networks for Computer Vision Applications (2013), 17–35.

[38] F. Glover, Tabu search part i, ORSA Journal on Computing 1 (3),(1989) 190-206.

[39] F. Glover, Tabu search part ii, ORSA Journal on Computing 2 (3) (1990) 4-32.

[40] S. B. Hamida and M. Schoenauer, *(ASCHEA): new results using adaptive segregational constraint handling*, in: Proceedings of the congress on evolutionary computation 2002 (CEC'2002), vol. 1, 2002, pp. 884-889.

[41] S. B. Hamida and M. Schoenauer, *Adaptive techniques for Evolutionary Topological Optimum Design*, in: I. Parmee (Ed.), Proceedings of the Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM2000), Springer-Verlag, University of Plymouth, Devon, UK, 2000, pp. 123-136.

[42] K. Hammouche, M. Diaf and P. Siarry, *A comparative study of various meta-heuristic techniques applied to the multilevel thresholding problem*, Engineering Applications of Artificial Intelligence 23(5) (2010), 676-688.

[43] R. Harrabi and E. B. Braiek, *Color image segmentation using multi-level thresholding approach and data fusion techniques: application in the breast cancer cells images*, EURASIP Journal on Image and Video Processing (2012).

[44] J. Heikkonen and M. Mantynen, *A computer vision approach to digit recognition on pulp bales*, Pattern Recognition Letters 17(4) (1996), 413-419.

[45] J. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, 1992.

[46] M.H. Horng, *Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation*, Expert Systems with Applications 38(11), 13 (2011), 13785-13791.

[47] E.T. Jaynes, *Information theory and statistical mechanics*, Physical Review, Series II 106(4),(1957), 620-630.

[48] E.J.N. Kapur, P.K. Sahoo and A.K.C Wong, *A new method for gray-level picture thresholding using the entropy of the histogram*, Computer Vision, Graphics, and Image Processing 29(3) (1985), 273-285.

[49] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005).

[50] D. Karaboga, B. Gorkemli, C. Ozturk and N. Karaboga *A comprehensive survey: artificial bee colony (ABC) algorithm and applications*, Artificial Intelligence Review 42 (1), Springer Netherlands (2014), 21–57.

[51] D. Karaboga and B. Akay, *PID Controller Design by using Artificial Bee Colony, Harmony Search and The Bees Algorithms*, Proceedings of the Institution of Mechanical Engineers, Part I, Journal of Systems and Control Engineering 224 (I7) (2010), 869–883.

[52] D. Karaboga and B. Basturk, *Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems*, in: LNAI 4529: IFSA'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 789–798.

[53] D. Karaboga and B. Akay, *A Modified Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Problems*, Applied Soft Computing 11 (3) (2011), 3021–3031.

[54] D. Karaboga and B. Basturk, *On the performance of artificial bee colony (ABC) algorithm*, Applied Soft Computing 8 (1) (2008), 687-697.

[55] D. Karaboga and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm*, Journal of Global Optimization 39 (3) (2007), 459–471.

[56] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, *Optimization by simulated annealing*, Science 220 (4598) (1983), 671–680.

[57] S. Kukkonen and J. Lampinen, *Constrained real-parameter optimization with generalized differential evolution*, in: IEEE congress on evolutionary computation 2006 (CEC 2006), 2006, pp. 207-214.

[58] M. H. Kashan, N. Nahavandi and A. H. Kashan, *DisABC: A new artificial bee colony algorithm for binary optimization*, Applied Soft Computing 12 (1) (2012), 342–352.

[59] J. Kennedy and R. C. Eberhart, *Particle swarm optimization*, In Proceedings of the 1995 IEEE international conference on neural networks (pp. 1942-1948). Piscataway, NJ: IEEE Service Center.

[60] O. Kisi , C. Ozkan and B. Akay *Modelling discharge-sediment relationship using neural networks with artificial bee colony algorithm*, Journal of Hydrology 428–429, Springer Netherlands (2012), 94–103.

[61] M.S. Kiran, H. Işcan and M. Gündüz, *The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem*, Neural Computing and Applications 23 (1) (2012), 9–21.

[62] M. K. Marichelvam, *An improved hybrid Cuckoo Search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems*, International Journal of Bio-Inspired Computation, 4 (4) (2012), 200–205.

[63] V. V. D. Melo and G. L. C. Carosio, *Investigating multi-view differential evolution for solving constrained engineering design problems*, Expert Systems with Applications, 40 (9) (2013), 3370-3377.

[64] V. V. D. Melo and G. L. C. Carosio, *Evaluating differential evolution with penalty function to solve constrained engineering problems*, Expert Systems with Applications, 39(9) (2012), 7860-7863.

[65] E. Mezura-Montes and O. Cetina-Domínguez, *Exploring promising regions of the search space with the scout bee in the artificial bee colony for constrained optimization*, Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE'2009), ASME Press Series, 19 (2009), 253–260.

[66] E. Mezura-Montes and O. Cetina-Domínguez, *Empirical analysis of a modified Artificial Bee Colony for constrained numerical optimization*, Applied Mathematics and Computation, 218 (22) (2012), 10943–10973.

[67] E. Mezura-Montes and C. A. Coello Coello, *Constraint-Handling in Nature-Inspired Numerical Optimization: Past, Present and Future*, Swarm and Evolutionary Computation, 1 (4) (2011), 173-194.

[68] E. Mezura-Montes, C. A. Coello Coello and E. I. Tun-Morales, *Simple Feasibility Rules and Differential Evolution for Constrained Optimization*, In: MICAI 2004: Advances in Artificial Intelligence (R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, H. Sossa, eds.), Lecture Notes in Computer Science, Volume 2972, 2004, pp. 707-716.

[69] E. Mezura-Montes, J. Velázquez-Reyes and C. A. Coello Coello, *Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization*, In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005), ACM Press, New York, 2005, pp. 225-232.

[70] A. W. Mohamed and H. Z. Sabry, *Constrained optimization based on modified differential evolution algorithm*, Information Sciences, 194 (2012), 171-208.

[71] H.F. Ng, *Automatic thresholding for defect detection*, Pattern Recognition Letters 27(14), (2006), 1644-1649.

[72] N. Otsu, *A threshold selection method for grey level histograms*, IEEE Transactions on Systems, Man and Cybernetics 9(1) (1979), 62-66

[73] M. Portes de Albuquerque, I.A. Esquef, A.R.Gesualdi Mello, *Image thresholding using Tsallis entropy*, Pattern Recognition Letters 25(9) (2004), 1059-1065

[74] R.V. Rao, V.J. Savsani and D.P. Vakharia, *Teachinglearning-based optimization: A novel method for constrained mechanical design optimization problems*, Computer-Aided Design 43 (2) (2011), 303-315.

[75] K. Rasheed, *An Adaptive Penalty Approach for Constrained Genetic Algorithm Optimization*, in: J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, R. L. Riolo (Eds.), Proceedings of the Third Annual Genetic Programming Conference, Morgan Kaufmann Publishers, San Francisco, California, 1998, pp. 584-590.

[76] T. P. Runarsson and X. Yao, *Stochastic ranking for constrained evolutionary optimization*, IEEE Transactions on Evolutionary Computation, 4 (3), (2000), 284-294.

[77] T. P. Runarsson and X. Yao, *Search biases in constrained evolutionary optimization*, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 35 (2), (2005), 233-243.

[78] X. Li and M. Yin, *Self-adaptive constrained artificial bee colony for constrained numerical optimization*, Neural Computing and Applications, 24 (3–4) (2014), 723–734.

[79] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A.C. Coello and K. Deb, *Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization*, Technical Report, Nanyang Technological University, Singapore (2006).

[80] J.J. Liang, A. K. Qin and P. N. Suganthan, *Comprehensive learning particle swarm optimizer for global optimization of multimodal functions*, IEEE Trans. on Evolutionary Computation 10(3) (2006), 67-82.

[81] Y.F. Liu and S.Y. Liu, *A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem*, Applied Soft Computing, 13 (3) (2013), 1459–1463.

[82] H. Liu, Z. Cai and Y. Wang *Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization*, Applied Soft Computing, 10 (2) (2010), 629-640.

[83] D. T. Pham, E. Koc, A. Ghanbarzadeh, S. Otri, S. Rahim, M. Zaidi, *The bees algorithm a novel tool for complex optimisation problems*, In IPROMS 2006 proceeding 2nd international virtual conference on intelligent production machines and systems (2006), pp. 454-459.

[84] P. Sahoo, C. Wilkins and J. Yeager, *Threshold Selection Using Renyi's Entropy*, Pattern Recognition 30(1) (1997), 71-84.

[85] S. Sarkar, G.R. Patra and S. Das, *A differential evolution based approach for multilevel image segmentation using minimum cross entropy thresholding*, in: Proceedings of the Second international conference on Swarm, Evolutionary, and Memetic Computing - Volume Part I, pp. 51-58 (2011).

[86] P.D. Sathya and R. Kayalvizhi, *Optimal multilevel thresholding using bacterial foraging algorithm*, Expert Systems with Applications 38(12) (2011), 15549-15564.

[87] P.D. Sathya and R. Kayalvizhi, *Modified bacterial foraging algorithm based multilevel thresholding for image segmentation*, Engineering Applications of Artificial Intelligence 24(4) (2011), 595-615.

[88] M. Sezgin and B. Sankur, *Survey over image thresholding techniques and quantitative performance evaluation*, Journal of Electronic Imaging 13(1) (2004), 146-165.

[89] A. Singh and S. Sundar, *An artificial bee colony algorithm for the minimum routing cost spanning tree problem*, Soft Computing 15 (12) (2013), 2489–2499.

[90] A. E. Smith and D. W. Coit, *Constraint Handling TechniquesPenalty Functions*, in: T. Bäck, D. B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Oxford University Press and Institute of Physics Publishing, 1997.

[91] T. K. Sharma and M. Pant, *Enhancing the food locations in an artificial bee colony algorithm*, Soft Computing 17 (10) (2013), 1939–1965.

[92] K. Sörensen and F. W. Glover, *Metaheuristics* , In: Encyclopedia of Operations Research and Management Science (Gass, SaulI. and Fu, MichaelC., eds.), Springer US, 2013, pp. 960–970.

[93] P. R. Srivastava, A. Varshney, P. Nama and X-S. Yang, *Software test effort estimation: a model based on cuckoo search*, International Journal of Bio-Inspired Computation 4 (5) (2012), 278–285.

[94] R. Storn and K. Price , *Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces*, Journal of Global Optimization 11 (4) (1997), 341–359.

[95] C. Sun , J. Zeng and J. Pan , *An improved vector particle swarm optimization for constrained optimization problems*, Information Sciences 181(6) (2011), 1153-1163.

[96] W.Y. Szeto, Y. Wu and Sin C. Ho, *An artificial bee colony algorithm for the capacitated vehicle routing problem*, European Journal of Operational Research 215 (1) (2011), 126–135.

[97] B. Shaw, S. Ghoshal and V. Mukherjee, *Solution of Economic Load Dispatch Problems by a Novel Seeker Optimization Algorithm*, International Journal on Electrical Engineering and Informatics 3 (1) (2011), 26–42.

[98] Y. Shi and R. Eberhart, *Empirical study of particle swarm optimization*, Proc. of the Congress on Evolutionary Computation,1999, 3: 1945-1950.

[99] H.-P. Schwefel, *Numerical Optimization of Computer Models*, Wiley & Sons, Great Britain, 1981.

[100] E-G. Talbi, Metaheuristics: from design to implementation, Wiley, ISBN: 978-0-470-27858-1, 624 pages, 2009.

[101] N. Taspinar, D. Karaboga, M. Yildirim and B. Akay, *PAPR reduction using artificial bee colony algorithm in OFDM systems*, Turkish Journal of Electrical Engineering & Computer Sciences 19 (2011), 47–58.

[102] M. Tuba, I. Brajevic and R. Jovanovic, *Hybrid Seeker Optimization Algorithm for Global Optimization*, Applied Mathematics and Information Sciences 7 (3) (2013), 867-875.

[103] M. Črepinšek, L. Shih-Hsi and M. Mernik, *Exploration and exploitation in evolutionary algorithms: A survey*, ACM Computing Surveys (CSUR) **45** (3) (2013), 1–33.

[104] H. Yan, *Unified formulation of a class of optimal image thresholding techniques* ,Pattern Recognition 29(12) (1996), 2025-2032.

[105] X-S. Yang, *Firefly algorithms for multimodal optimization* ,In: Stochastic Algorithms: Foundations and Applications (O. Watanabe, T. Zeugmann, eds.), Lecture Notes in Computer Science, Volume 5792, Springer Berlin Heidelberg, 2009, pp. 169–178.

[106] X-S. Yang, *Firefly algorithm: recent advances and applications* , Int. J. of Swarm Intelligence **1**(1) (2013), 36–50.

[107] X-S. Yang, *Nature-Inspired Metaheuristic Algorithms* , Luniver Press, United Kingdom, 2010.

[108] X-S. Yang and S. Deb, *Cuckoo search via Lévy flights* , Proc. of World Congress on Nature & Biologically Inspired Computing, 2009, pp. 210–214.

[109] X-S. Yang and S. Deb, *Engineering Optimisation by Cuckoo Search* ,Int. J. of Mathematical Modelling and Numerical Optimisation 1 (4) 2010, 330–343.

[110] X-S. Yang, C. Huyck, M. Karamanoglu and N. Khan, *True global optimality of the pressure vessel design problem: a benchmark for bio-inspired optimisation algorithms* , International Journal of Bio-Inspired Computation 5 (6) 2013, 329-335.

[111] X-S. Yang, *Engineering optimizations via nature-inspired virtual bee algorithms* , In: Artificial intelligence and knowledge engineering applications: A bioinspired approach, LNCS Vol. 3562, 2005, pp. 317-323, Springer.

[112] X-S. Yang, *Nature-inspired metaheuristic algorithms: success and new challenges* , Journal of Computer Engineering and Information Technology 1 (1), 2012.

[113] X. Yao, Y. Liu, and G. Lin, *Evolutionary programming made faster* , IEEE Trans. Evol. Comput. 3 (2) 1999, 82–102.

[114] W.C. Yeh and T.J. Hsieh, *Solving reliability redundancy allocation problems using an artificial bee colony algorithm*, Computers & Operations Research 38 (110) (2011), 1465–1473.

[115] A. R. Yildiz, *Hybrid Taguchi-harmony search algorithm for solving engineering optimization problems*, International Journal of Industrial Engineering: Theory, Applications and Practice 15 (3) (2008), 286-293.

[116] J.C. Yen, F.J. Chang and S. Chang, *A new criterion for automatic multilevel thresholding*, IEEE Transaction on Image Processing 4(3)(1995), 370-378.

[117] A. R. Yildiz, *Comparison of evolutionary based optimization algorithms for structural design optimization*, Engineering Applications of Artificial Intelligence 26 (1) (2013), 327-333.

[118] A. R. Yildiz, *A new hybrid bee colony optimization approach for robust optimal design and manufacturing*, Applied Soft Computing 13 (5) (2013), 2906-2912.

[119] A. R. Yildiz, *A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations*, Applied Soft Computing 13 (3) (2013), 1561-1566.

[120] P.Y. Yin, *Multilevel minimum cross entropy threshold selection based on particle swarm optimization*, Applied Mathematics and Computation 184(2) (2007), 503-513.

[121] Y. Zhou, H. Zheng, Q. Luo and J. Wu, *An improved Cuckoo Search Algorithm for Solving Planar Graph Coloring Problem*, Applied Mathematics & Information Sciences 7 (2) (2013), 785–792.

[122] G. Zhu and S. Kwong, *Gbest-guided artificial bee colony algorithm for numerical function optimization*, Applied Mathematics and Computation 217 (7) (2010), 3166-3173.

Dodatak A

Improvements of some population-based metaheuristics for constrained optimization problems

Many constrained optimization problems are hard optimization problems. As dimensionality of the problems grows, complexity of classical algorithms, used to solve them, grows exponentially. Depending on the nature of the problem, often it is possible to obtain only approximate solutions. Many hard optimization problems are still unsolvable by classical algorithms despite of improvements of computers performances.

However, usage of computers for simulations of certain processes in nature, provides a greater opportunity for solving hard optimization problems. Following the steps of some natural phenomenons, leads to development of nature inspired algorithms. Their implementation in computer programs and verification of their results shows that these algorithms can be successfully applied to solve many hard optimization problems in a reasonably practical time.

The development of algorithms for solving optimization problems goes into two directions. The first group of algorithms are deterministic algorithms. They always produce the same output for the given input for every execution. These algorithms involve a considerable amount of numerical calculations. Also, the quality of their results depends on the ability to adjust the problem to the given algorithm.

The second group of algorithms are heuristic methods which, at least in a part of the execution, use a random component. Heuristic methods

do not guarantee that they will produce optimal and feasible results. On the other hand, their advantage is that they are able to solve optimization problems in relatively short time and these methods can be adjusted to different optimization problems.

The main disadvantage of heuristics is that they are developed to solve a certain problem and they depend on the characteristics of the problem they are solving. With an intention to generalize the heuristic algorithms, a new class of algorithms, called metaheuristics, are emerged. Metaheuristics require relatively few assumptions of the problem and they can be applied to a broader set of problems. These algorithms are mostly inspired by nature and they are being developed rapidly in the last two decades.

In this thesis some improved variants of some prominent population-based metaheuristics for constrained optimization problems are presented. The metaheuristics which are considered are: artificial bee colony algorithm (ABC), cuckoo search (CS), firefly algorithm (FA) and seeker optimization algorithm (SOA). It has been observed that an enhancement of an optimization algorithm can be achieved in many ways: by modifying the search equation, by combining search equations from different metaheuristics, by introducing new selection operators and by fine tuning of control parameters of an algorithm. The proposed improved variants of the population-based metaheuristcs are tested on a large set of standard benchmark problems, as well as for solving real-life problems which are considered to be constrained optimization problems.

The basic concepts related to constrained optimization problems, the methods which are used to solve these types of problems and the detailed description of the considered population based metaheuristcs are presented in the first Chapter of this thesis.

In the second Chapter of this dissertation an improved variant of the artificial bee colony algorithm is presented. The proposed crossover-based artificial bee colony (CB-ABC) algorithm introduces five modifications to the ABC algorithm proposed to solve constrained optimization problems. In order to improve the exploitation abilities of ABC, two different modified ABC search operators are used in employed and onlooker phases, and crossover operator is used in scout phase instead of random search. The remaining two modifications are usage of dynamic tolerance for handling equality constraints and improved boundary constraint handling method.

The proposed CB-ABC algorithm is tested on a set of twenty-four well-known benchmark functions and four widely used engineering design problems. The obtained results demonstrated a better performance of the CB-

ABC algorithm than the approaches based on the ABC, and the most competitive approaches which were previously compared with the ABC algorithm with respect to the quality and robustness of the results with improved convergence speed.

The Chapter is based on the paper:

- I. Brajevic, *Crossover-based artificial bee colony algorithm for constrained optimization problems*, Neural Computing and Applications, ISSN 0941-0643, DOI: 10.1007/s00521-015-1826-y, 15 p., 2015.

In the third Chapter of this dissertation hybridization of the seeker optimization algorithm with the artificial bee colony algorithm is presented. The proposed algorithm, called hybrid seeker optimization algorithm (HSO), is employed to solve global optimization problems. The HSO algorithm at certain stages modifies seeker's positions by search formulas from the ABC algorithm and also modifies the inter-subpopulation learning phase by using the binomial crossover operator.

The proposed approach was tested on the complete set of twenty-three benchmark functions. Comparisons show that the proposed algorithm outperforms the SOA, ABC, and four other state-of-the-art algorithms in terms of the quality of the results, as well as robustness on most of the test functions. Also, the proposed hybrid algorithm successfully overcomes SOA algorithm's tendency to prematurely converge to local optima for multimodal functions.

The Chapter is based on the paper:

- M. Tuba, I. Brajevic and R. Jovanovic, *Hybrid Seeker Optimization Algorithm for Global Optimization*, Applied Mathematics and Information Sciences 7 (3) (2013), 867-875.

In the fourth Chapter of this thesis, an enhanced firefly algorithm (E-FA) is proposed to solve structural optimization problems with mixed variables. Two modifications related to the constraint handling method based on three feasibility rules and the geometric progression reduction scheme mechanism are introduced in order to enhance its performance in the constrained search space. The E-FA is tested on four classical structural optimization problems. Obtained results show that the proposed approach was very competitive in the considered problems, outperforming the original firefly algorithm in most of the problems.

The Chapter is based on the paper:

- I. Brajevic and J. Ignjatović, *An enhanced firefly algorithm for mixed variable structural optimization problems*, Facta Universitatis: Series Mathematics and Informatics, 2015, (accepted for publication).

The fifth Chapter of this thesis presents the adaptations of the cuckoo search algorithm and firefly algorithm to solve the multilevel thresholding problem. This problem is often treated as a problem of optimization of an objective function. In this Chapter two different objective functions, Kapurs maximum entropy thresholding function and multi Otsu between-class variance, were used on six standard test images with known optima from exhaustive search, up to five threshold points. Differential evolution (DE) and particle swarm optimization (PSO) algorithms have also been implemented for the purpose of comparison with the CS and FA.

The experimental results show that, for both thresholding criteria, as the number of desired thresholds increases, the superiority of the CS and FA over the PSO and DE becomes more pronounced, both in terms of precision and robustness, with a small advantage of the FA. The computational time analysis shows that among these algorithms, the FA converges the most quickly. Also, the CPU times for each of these metaheuristcs are reasonable and grow at linear rate as the problem dimension increases, as opposed to the exhaustive search where the growth is exponential.

The Chapter is based on the paper:

- I. Brajevic and M. Tuba, *Cuckoo Search and Firefly Algorithm Applied to Multilevel Image Thresholding*, In: Cuckoo Search and Firefly Algorithm (X.S. Yang,eds.), Studies in Computational Intelligence, Volume 516, Springer International Publishing, 2014, pp. 115-139.

For each proposed improved population-based algorithm in this thesis, the tables of comparative results are provided. Data is compared according to the best optimization results, the best possible result (if it is known), the average values of obtained results, the standard deviation values and the number of evaluations.

Results demonstrated in this theses contribute to the attitude that the population-based metaheuristics can be successfully applied for solving a wide set of different, hard optimization problems. Each of enhanced population-based metaheuristics, presented in this thesis, is a contribution to the development of the theory of the corresponding original population-based algorithm, as well as to the development of the theory of population-based metaheuristics in general.

All of the presented improved population-based metaheuristics in this thesis are published in prestigious international journals. Their influence is confirmed by a considerable number of citations in international scientific publications.

Dodatak B

Biografija

Ivona Brajević je rođena 28.09.1980. godine u Kruševcu. Osnovnu školu završila je u Beogradu, a srednje obrazovanje stekla je u Devetoj beogradskoj gimnaziji.

Osnovne i master studije završila je na Matematičkom fakultetu Univerziteta u Beogradu. Na doktorskim studijama na Prirodno-matematičkom fakultetu Univerziteta u Nišu, departman za računarske nauke, položila je sve ispite predviđene nastavnim planom i programom sa prosečnom ocenom 10.

Nakon završenih osnovnih studija radila je kao profesor matematike u Osmoj beogradskoj gimnaziji. Od 2008. godine radi na Visokoj školi za poslovnu ekonomiju i preduzetništvo.

Objavila je preko 15 naučnih radova u vodećim međunarodnim časopisima indeksiranim na SCI listi (Thompson Reuters ISI), Scopus-u i/ili MathSciNet-u, kao i na međunarodnim konferencijama indeksiranim u Web of Science, Scopus i/ili IEEE Xplore. Takođe je radila veći broj recenzija radova za vodeće međunarodne časopise (indeksirane na SCI listi) i međunarodne konferencije.

Angažovana je na projektu III-44006 iz programa integralnih interdisciplinarnih istraživanja, Ministarstva za obrazovanje, nauku i tehnološki razvoj Republike Srbije.

Dodatak C

Dokumentacija za disertaciju

Прилог 4/1

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:	
Идентификациони број, **ИБР**:	
Тип документације, **ТД**:	Монографска
Тип записа, **ТЗ**:	Текстуални, графички
Врста рада, **ВР**:	Докторска дисертација
Аутор, **АУ**:	Ивона Брајевић
Ментор, **МН**:	Јелена Игњатовић
Наслов рада, **НР**:	Побољшања неких популационих метахеуристика за решавање оптимизационих проблема са ограничењима
Језик публикације, **ЈП**:	Српски
Језик извода, **ЈИ**:	Енглески
Земља публиковања, **ЗП**:	Република Србија
Уже географско подручје, **УГП**:	Република Србија
Година, **ГО**:	2015
Издавач, **ИЗ**:	Ауторски репринт
Место и адреса, **МА**:	Ниш, Вишеградска 33
Физички опис рада, **ФО**:	118 страна
Научна област, **НО**:	Рачунарске науке
Научна дисциплина, **НД**:	Вештачка интелигенција
Предметна одредница/кључне речи, **ПО**:	Популационе метахеуристике, оптимизациони проблеми
УДК	004.023, 004.832.2
Чува се, **ЧУ**:	Библиотека
Важна напомена, **ВН**:	
Извод, **ИЗ**:	У докторској дисертацији су представљене побољшане варијанте неких истакнутих популационих метахеуристика за решавање оптимизационих проблема са ограничењима. Посебно су разматране: метахеуристика базирана на пчелињим колонијама, алгоритам свитаца, претрага кукавице и метахеуристика базирана на популацији људске групе. Уочено је да се побољшња могу постићи модификацијом једначина претраге или комбиновањем једначина претраге различитих метахеуристика, коришћењем нових оператора селекције, као и финим подешавањима параметара који контролишу извршавање алгоритма. Предложене побољшане варијанте популационих алгоритама су тестиране на великом скупу стандардних бенчмарк проблема , као и за решавање неких реалних проблема који представљају оптимизационе проблеме са ограничењима.
Датум прихватања теме, **ДП**:	06. 04. 2015.
Датум одбране, **ДО**:	
Чланови комисије, **КО**: Председник:	
Члан-ментор:	
Члан:	
Члан:	

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Docyment type, **DT**:	Monograph
Type of record, **TR**:	Textual / graphic
Contents code, **CC**:	Doctoral dissertation
Author, **AU**:	Ivona Brajević
Mentor, **MN**:	Jelena Ignjatović
Title, **TI**:	Improvements of Some Population-Based Metaheuristics for Constrained Optimization Problems
Language of text, **LT**:	Serbian
Language of abstract, **LA**:	English
Country of publication, **CP**:	Serbia
Locality of publication, **LP**:	Serbia
Publication year, **PY**:	2015
Publisher, **PB**:	Author's reprint
Publication place, **PP**:	Niš, Višegradska 33
Physical description, **PD**:	118 pages; graphic representations
Scienfitic field, **SF**:	Computer Science
Scienfitic discipline, **SD**:	Artifficial intelligence
Subject/Key words**, S/KW**:	Population-Based Metaheuristics, Optimization Problems
UC	004.023, 004.832.2
Holding data, **HD**:	Library
Note, **N**:	
Abstract, **AB**:	In this thesis some improved variants of some prominent population-based metaheuristics for constrained optimization problems are presented. The metaheuristics which are considered are: artificial bee colony algorithm (ABC), cuckoo search (CS), firefly algorithm (FA) and seeker optimization algorithm (SOA). It has been observed that an enhancement of an optimization algorithm can be achieved in many ways: by modifying the search equation, by combining search equations from different metaheuristics, by introducing new selection operators and by fine tuning of control parameters of an algorithm. The proposed improved variants of the population-based metaheuristcs are tested on a large set of standard benchmark problems, as well as for solving real-life problems which are considered to be constrained optimization problems.
Accepted by the Scientific Board on, **ABS**:	April 6th, 2015
Defended on, **DE**:	
Defended Board , **DB**: President:	
Member-mentor:	
Member:	
Member:	

Универзитет у Нишу

ИЗЈАВА О АУТОРСТВУ

Изјављујем да је докторска дисертација, под насловом

ПОБОЉШАЊА НЕКИХ ПОПУЛАЦИОНИХ МЕТАХЕУРИСТИКА ЗА РЕШАВАЊЕ ОПТИМИЗАЦИОНИХ ПРОБЛЕМА СА ОГРАНИЧЕЊИМА

која је одбрањена на Природно-математичком факултету Универзитета у Нишу:

- резултат сопственог истраживачког рада;
- да ову дисертацију, ни у целини, нити у деловима, нисам пријављивао/ла на другим факултетима, нити универзитетима;
- да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица.

Дозвољавам да се објаве моји лични подаци који су у вези са ауторством и добијањем академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада и то у каталогу Библиотеке, Дигиталном репозиторијуму Универзитета у Нишу, као и у публикацијама Универзитета у Нишу.

У Нишу, 19.06.2015.

Аутор дисертације: Ивона Брајевић

Потпис аутора дисертације

Ивона Брајевић

Универзитет у Нишу

ИЗЈАВА О ИСТОВЕТНОСТИ ШТАМПАНОГ И ЕЛЕКТРОНСКОГ ОБЛИКА ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Име и презиме аутора: Ивона Брајевић

Наслов дисертације:

ПОБОЉШАЊА НЕКИХ ПОПУЛАЦИОНИХ МЕТАХЕУРИСТИКА ЗА РЕШАВАЊЕ ОПТИМИЗАЦИОНИХ ПРОБЛЕМА СА ОГРАНИЧЕЊИМА

Ментор: др Јелена Игњатовић

Изјављујем да је штампани облик моје докторске дисертације истоветан електронском облику који сам предао/ла за уношење у **Дигитални репозиторијум Универзитета у Нишу**.

У Нишу, 19.06.2015.

Потпис аутора дисертације

Ивона Брајевић

Универзитет у Нишу

ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Никола Тесла“ да, у Дигитални репозиторијум Универзитета у Нишу, унесе моју докторску дисертацију, под насловом:

ПОБОЉШАЊА НЕКИХ ПОПУЛАЦИОНИХ МЕТАХЕУРИСТИКА ЗА РЕШАВАЊЕ ОПТИМИЗАЦИОНИХ ПРОБЛЕМА СА ОГРАНИЧЕЊИМА

Дисертацију са свим прилозима предао/ла сам у електронском облику, погодном за трајно архивирање.

Моју докторску дисертацију, унету у Дигитални репозиторијум Универзитета у Нишу, могу користити сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons), за коју сам се одлучио/ла.

1. Ауторство (**CC BY**)
2. Ауторство - некомерцијално (**CC BY-NC**)
3. Ауторство - некомерцијално - без прераде (**CC BY-NC-ND**)
4. Ауторство - некомерцијално - делити под истим условима (**CC BY-NC-SA**)
5. Ауторство - без прераде (**CC BY-ND**)
6. Ауторство - делити под истим условима (**CC BY-SA**)

(Молимо да подвучете само једну од шест понуђених лиценци; опис лиценци дат је у наставку текста).

У Нишу, 19.06.2015.

Аутор дисертације: Ивона Брајевић

Потпис аутора дисертације

Ивона Брајевић