- UNIVERZITET U BEOGRADU -

MATEMATIČKI FAKULTET

NEBOJSA V. BAČANIN DŽAKULA

UNAPREĐENJE HIBRIDIZACIJOM METAHEURISTIKA INTELIGENCIJE ROJEVA ZA REŠAVANJE PROBLEMA GLOBALNE OPTIMIZACIJE

DOKTORSKA DISERTACIJA

BEOGRAD - JUN 2015.

- UNIVERSITY OF BELGRADE -

FACULTY OF MATHEMATICS

NEBOJSA V. BAČANIN DŽAKULA

IMPROVEMENT BY HYBRIDIZATION OF SWARM INTELLIGENCE METAHEURISTICS FOR SOLVING GLOBAL OPTIMIZATION PROBLEMS

PhD THESIS

BELGRADE - JUNE 2015.

MENTOR:

dr Milan Tuba, vanredni profesor

UNIVERZITET U BEOGRADU, MATEMATIČKI FAKULTET

ČLANOVI KOMISIJE:

dr Miodrag Živković, redovni profesor

UNIVERZITET U BEOGRADU, MATEMATIČKI FAKULTET

dr Đorđe Dugošija, redovni profesor u penziji

UNIVERZITET U BEOGRADU, MATEMATIČKI FAKULTET

dr Xin-She Yang, profesor

MIDDLESEX UNIVERSITY, LONDON, UK

Datum odbrane:

Mojoj porodici i prijateljima

Sadržaj

Rezime

Teški optimizacioni problemi, nerešivi u prihvatljivom vremenu izvršavanja determinističkim matematičkim metodama, uspešno se poslednjih godina rešavaju populacionim stohastičkim metaheuristikama, među kojima istaknutu klasu predstavljaju algoritmi inteligencije rojeva. U ovom radu razmatra se unapređenje metaheuristika inteligencije rojeva pomoću hibridizacije. Analizom postojećih metaheuristika u određenim slučajevima uočeni su nedostaci i slabosti u mehanizmima pretrage prostora rešenja koji pre svega proističu iz samog matematičkog modela kojim se simulira proces iz prirode kao i iz nedovoljno usklađenog balansa između intenzifikacije i diversifikacije.

U radu je ispitivano da li se postojeći algoritmi inteligencije rojeva za globalnu optimizaciju mogu unaprediti (u smislu dobijanja boljih rezultata, brže konvergencije, veće robustnosti) hibridizacijom sa drugim algoritmima. Razvijeno je i implementirano više hibridizovanih metaheuristika inteligencije rojeva. S obzirom da dobri hibridi ne nastaju „slučajnom" kombinacijom pojedinih funkcionalnih elemenata i procedura različitih algoritama, već su oni utemeljeni na sveobuhvatnom izučavanju načina na koji algoritmi koji se hibridizuju funkcionišu, kreiranju hibridnih pristupa prethodila je detaljna analiza prednosti i nedostataka posmatranih algoritma kako bi se napravila najbolja kombinacija koja nedostatke jednih neutrališe prednostima drugih pristupa.

Razvijeni hibridni algoritmi verifikovani su testiranjima na standardnim skupovima test funkcija za globalnu optimizaciju sa ograničenjima i bez ograničenja, kao i na poznatim praktičnim problemima. Upoređivanjem sa najboljim poznatim algoritmima iz literature pokazan je kvalitet razvijenih hibrida, čime je potvrđena i osnovna hipoteza ovog rada da se algoritmi inteligencije rojeva mogu uspešno unaprediti hibridizacijom.

Ključne reči: metaheuristike inteligencije rojeva, hibridni algoritmi, globalna optimizacija

Naučna oblast: Računarstvo

Uža naučna oblast: Veštačka inteligencija

UDK broj: 004 (004.8)

CERIF: P176, P170

Abstract

Hard optimization problems that cannot be solved within acceptable computational time by deterministic mathematical methods have been successfully solved in recent years by population-based stochastic metaheuristics, among which swarm intelligence algorithms represent a prominent class. This thesis investigates improvements of the swarm intelligence metaheuristics by hybridization. During analysis of the existing swarm intelligence metaheuristics in some cases deficiencies and weaknesses in the solution space search mechanisms were observed, primarily as a consequence of the mathematical model that simulates natural process as well as inappropriate balance between intensification and diversification.

The thesis examines whether existing swarm intelligence algorithms for global optimization could be improved (in the sense of obtaining better results, faster convergence, better robustness) by hybridization with other algorithms. A number of hybridized swarm intelligence metaheuristics were developed and implemented. Considering the fact that good hybrids are not created as a „random" combination of individual functional elements and procedures from different algorithms, but rather established on comprehensive analysis of the functional principles of the algorithms that are used in the process of hybridization, development of the hybrid approaches was preceded by thorough research of advantages and disadvantages of each involved algorithm in order to determine the best combination that neutralizes disadvantages of one approach by incorporating the strengths of the other.

Developed hybrid approaches were verified by testing on standard benchmark sets for global optimization, with and without constraints, as well as on well-known practical problems. Comparative analysis with the state-of-the-art algorithms from the literature demonstrated quality of the developed hybrids and confirmed the hypothesis that swarm intelligence algorithms can be successfully improved by hybridization.

Keywords: swarm intelligence metaheuristics, hybrid algorithms, global optimization

Scientific field: Computer Science

Scientific subfield: Artificial Intelligence

UDC number: 004 (004.8)

CERIF: P176, P170

Zahvalnica

Upućujem zahvalnost svima onima koji su mi svojom podrškom i iskrenim savetima pomagali svih ovih godina da postignem svoj cilj.

Posebnu zahvalnost dugujem svom mentoru, profesoru Milanu Tubi, na dugogodišnjoj saradnji, podršci i razumevanju. Mentor me je svih ovih godina nesebičnim deljenjem korisnih saveta, kako o tezi, tako i o životu, usmeravao ka postizanju svog cilja. Profesor Tuba me je izučio zanatu pisanju radova i zahvaljujući njemu sam učestovao na mnogim međunarodnim konferencijama i objavio radove u vrhunskim svetskim časopisima. Sećam se dobro svih onih noći koje smo profesor i ja provodili na Skype-u i radili na ko zna kojoj iteraciji rada koji treba da se pošalje. Profesor nikada nije štedio ni svoje reči, niti vreme da mi svaki problem objasni i razjasni do detalja, kako bi od mene napravio samostalnog istraživača.

Veliku zahvalnost dugujem članovima komisije, profesoru Miodragu Živkoviću i profesoru Đorđu Dugošiji na pažljivom čitanju teze i korisnim sugestijama koje su doprinele njenom boljem kvalitetu. Posebna mi je čast učešće u komisiji profesora Xin-She Yang-a sa Middlesex University u Velikoj Britaniji. Profesor Xin-She Yang je tvorac nekoliko algoritma inteligencije rojeva i jedan od najvećih svetskih autoriteta iz ove oblasti i njegovo veliko znanje pomoglo mi je da bolje razumem ovu vrlo zanimljivu i dinamičnu oblast.

Zahvalnost takođe dugujem kolegi Raki Jovanoviću, koji me je svojim savetima i sugestijama, usmeravao na pravi istraživački put, kao i svim drugim kolegama koji su verovali u mene sve ove godine.

Poslednje, ali ne i najmanje važno, zahvaljujem se svojoj porodici i prijateljima bez kojih ne bih uspeo da ostvarim svoj cilj.

Uvod

U poslednjih dvadesetak godina došlo je do razvoja niza populacionih stohastičkih metaheuristika za rešavanje teških optimizacionih problema, kako iz oblasti kombinatorne, tako i iz oblasti globalne optimizacije. Za NP [1] (Nondeterministic Polynomial-time) teške probleme obično je lako napisati algoritam baziran na kompletnoj pretrazi. Međutim, za izvršavanje ovakvog algoritma mogu biti potrebne hiljade godina. Zato se takvi problemi smatraju praktično nerešivima, pa je za njihovo rešavanje pogodno koristiti stohastičke metaheurističke metode koje ne garantuju da će dati optimalno rešenje, ali zato postižu zadovoljavajuće suboptimalne rezultate u razumnom vremenskom periodu.

Stohastičke populacione metaheuristike proces pretrage počinju od izvesnog broja slučajnih tačaka, kao Monte Karlo metod, ali se proces dalje vodi i usmerava nekakvim mehanizmom iz iteracije u iteraciju. Principi na kojima se bazira mehanizam kojim se usmerava pretraga može da bude inspirisan sistemima i procesima iz prirode. Rešenja evoluiraju ka optimalnom rešenju na osnovu poznavanja okruženja (vrednost funkcije podobnosti), istorije prethodnih stanja individualnih rešenja (memorija svake individue) i istorije stanja „susedstva" individua.

Jedan od najpoznatijih i najstarijih predstavnika ove klase su genetski algoritmi (GA), dok noviju grupu čine algoritmi inteligencije rojeva. Algoritmi inteligencije rojeva, inspirisani prirodnim biološkim sistemima, naročito kolonijama insekata, rade sa populacijom samoorganizujućih individua (agenata) koje inter-reaguju međusobno na lokalnom nivou, a na globalnom sa svojim okruženjem. Iako ne postoji centralna komponenta koja kontroliše i usmerava ponašanje pojedinaca, lokalne iterakcije između pojedinaca rezultuju nastankom globalno koordinisanog ponašanja.

Ovaj rad bavi se unapređenjima metaheuristika rojeva pomoću hibridizacije za re-

[1] U radu je pokušana razumna uniformna upotreba srpskih termina, uz početno navođenje engleskog originala. Međutim, s obzirom da se u radu spominje veliki broj algoritama, koji se standardno u literaturi nazivaju engleskim skraćenicama, i u radu su korišćene ove skraćenice, bez obzira na srpske termine

šavanje problema globalne optimizacije. Pažljivim proučavanjem algoritama inteligencije rojeva, u određenim slučajevima, uočene su slabosti i nedostaci, koji se pre svega odnose na mehanizme pretrage i na neodgovarajući balans između intenzifikacije i diversifikacije. Testiranjem je uvrđeno, a i u literaturi je razmatrano [1] da ovi populacioni algoritmi u mnogim slučajevima daju suboptimalna rešenja i zaglavljuju se u lokalnim optimumima.

Jedan od načina da se prevaziđu pomenuti problemi metaheuristika rojeva je hibridizacija. Hibridizacija ne treba da se bazira na slučajnoj kombinaciji algoritama koji se hibridizuju, već treba da bude ciljana i da se temelji na pažljivom proučavanju prednosti i nedostataka svakog algoritma. Nedostatak jednog algoritma (spora konvergencija, tendencija lakom zaglavljivanju u lokalni optimum, itd., u određenim uslovima) neutrališe se kombinovanjem sa strukturalnim delom drugog algoritma koji može da ispravi taj nedostatak.

Rad se sastoji iz uvoda, zaključka i pet poglavlja.

Posle uvoda, u prvom poglavlju prikazani su problemi optimizacije. Prikazane su vrste optimizacionih problema, data je njihova definicija i klase kompleksnosti. Takođe je prikazana i taksonomija optimizacionih metoda i algoritama.

U drugom poglavlju je dat pregled algoritama metaheurističke optimizacije. Prvo je data definicija i kategorizacija heurističkih metoda, kao osnova na kojima se baziraju metaheuristike. Zatim su prikazani teorijski koncepti i principi metaheuristika, kao i njihova klasifikacija. U nastavku poglavlja prikazane su metaheuristike koje su inspirisane prirodom i one koje nisu inspirisane prirodom.

Celo treće poglavlje posvećeno je metaheuristikama inteligencije rojeva. U ovom poglavlju je opisano nekoliko algoritama rojeva. Takođe su dati opisi bioloških sistema koje ovi algoritmi emuliraju, prikazane su teorijske osnove i principi svakog algoritma sa pseudo-kodom i dat je kratak pregled njihovih implementacija iz literature.

Četvrti deo se bavi hibridizacijom, kao načinom za unapređenje stohastičkih metaheuristika. U ovom poglavlju su prvo razmotreni opšti potencijalni nedostaci metaheuristika, a zatim i konkretni primeri koji su uočeni tokom ovog istraživanja.

Zatim su u pregledu memetskih i hibridnih algoritama navedeni njihovi principi rada, teorijske osnove i taksonomija.

U petom poglavlju izloženi su osnovni rezultati ovog rada, razmotreni su algoritmi koji su hibridizovani i uočeni su problemi koje oni imaju u određenim uslovima, kako s teorijskog, tako i s praktičnog aspekta. Svi hibridni algoritmi inteligencije rojeva koji su implementirani i testirani detaljno su opisani i prikazani. Uz prikazane eksperimentalne rezultate na benčmark i praktičnim problemima i komparativnu analizu sa drugim metaheuristikama iz savremene literature, data je detaljna diskusija o postignutim rezultatima.

Konačno, u zaključku su data završna razmatranja, gde su predloženi budući pravci istraživanja u ovoj oblasti.

Implementirano je više algoritama rojeva, kako osnovnih, tako i hibridizovanih. U radu je prikazano sedam hibridnih implementacija, kako za opšte benčmark probleme, tako i za primenu na praktične teške optimizacione probleme. Dobijeni rezultati su upoređivani sa najboljim rezultatima iz savremene literature. Na osnovu rezultata istraživanja, zaključuje se da su hibridne metaheuristike dale bolje rezultate od osnovnih implementacija.

1 Problemi optimizacije

Optimizacija može da se definiše kao postupak kojim se nalazi minimum ili maksimum neke funkcije. Optimizacija se koristi praktično svuda. Tako na primer, optimizuje se saobraćaj, sistemi navodnjavanja, raspored letenja, finansijski portfolijo, proces distribucije električne energije, itd.

Cilj procesa optimizacije je da se pronađe rešenje koje je ili optimalno, ili blizu optimalnog u odnosu na postavljene ciljeve i moguća ograničenja. Proces optimizacije može da se podeli na nekoliko faza (koraka) koji se izvršavaju sekvencijalno, tj. jedan za drugim [2]: prepoznavanje i definisanje problema, konstruisanje i rešavanje modela problema, implementacija i evaluacija rešenja.

Prilikom rešavanja problema iz bilo kog domena, donosioci odluke nisu u poziciji da izaberu bilo koju od ponuđenih alternativa rešenja problema, zbog činjenice da obično postoje ograničenja koja ograničavaju broj raspoloživih alternativna. Generalno posmatrano, problemi optimizacije imaju sledeće karakteristike [2]:

- raspoloživost različitih alternativa za rešavanje problema;
- dodatna ograničenja sužavaju broj mogućih alternativa;
- svaka alternativa može da ima drugačiji uticaj na kriterijum evaluacije i
- funkcija evaluacije koja se primenjuje na alternativu za rešenje problema opisuje efekat koji svaka alternativa ima.

Da bi se počelo sa procesom optimizacije, prvo je potrebno da se identifikuje funkcija cilja, kao kvantitativna mera performansi optimizacionog problema. Funkcija cilja može da bude na primer vrednost funkcije, profit, vreme, potrošena energija, ili bilo koja druga mera ili njihova kombinacija koja se prikazuje kao jedan broj [3].

Funkcija cilja zavisi od karakteristika optimizacionog problema koje se nazivaju varijable ili promenljive. Cilj je da je pronađu vrednosti varijabli koje optimizuju datu funkciju cilja. U velikom broju problema optimizacije, varijable mogu da imaju samo određene vrednosti. Tako na primer, gustina elektrona u molekulu ili kamatna stopa ne smeju da budu negativne. Veliki broj optimizacionih problema uključuje i

ograničenja.

Proces kojim se utvrđuje funkcija cilja, varijable i eventualna ograničenja datog problema zove se modeliranje [3]. Modeliranje je osetljiv proces i treba voditi računa o formulaciji modela. Ako je model previše jednostavan, on neće realno da modelira sve karakteristike problema na koji se odnosi. U suprotnom, ukoliko je model previše složen, ni jedna metoda neće moći uspešno da ga reši. Kada se model formuliše primenjuje se određena optimizaciona metoda.

U rešavanju problema optimizacije, potrebno je izabrati alternativu koja zadovoljava sva ograničenja i koja minimizuje (ili maksimizuje) funkciju cilja. U nastavku se bez smanjenja opštosti kao optimizacija uvek razmatra minimizacija.

Matematička formulacija problema minimizacije (za specijalni slučaj R^n koji se jedino i razmatra u ovom radu) može da se prikaže kao:

$$\begin{aligned} &\text{za dato } f : R^n \to R \\ &\text{pronaći } x^* \in R^n, \text{ tako da bude zadovoljeno} \\ &\forall x \in R^n,\ f(x^*) \leq f(x) \end{aligned} \tag{1.1}$$

U prikazanim jednačinama, R^n se naziva prostor pretrage funkcije f, a kako je potrebno da se pronađe skup odgovarajućih vrednosti varijabli, ovaj prostor se u literaturi još naziva i prostor varijabli. Potencijalno rešenje problema je $x \in R^n$, a optimalno rešenje je označeno sa x^*. Varijabla n označava broj dimenzija prostora pretrage, a u slučaju problema optimizacije funkcije broj varijabli.

1.1 Klasifikacija problema optimizacije

Data je klasifikacija problema optimizacije na osnovu tri kriterijuma: prema tipu varijabli, prema kriterijumu da li optimizacioni problem ima, ili nema ograničenja i prema broju funkcija cilja.

1.1.1 Podela prema tipu varijabli

Prema tipu varijabli problemi optimizacije mogu biti kombinatorni ili globalni. Kombinatorni problemi uzimaju varijable koje imaju diskretnu vrednost, dok globalni koriste realne (neprekidne) varijable.

1.1.1.1 Kombinatorna optimizacija

Kombinatorna ili diskretna optimizacija obuhvata probleme čije varijable imaju diskretnu (celobrojnu) vrednost. Kombinatorna optimizacija je matematička disciplina koja se bavi problemima pronalaženja ekstremnih vrednosti funkcije koja je definisana na konačnom ili prebrojivom skupu. Kao i kod svih drugih optimizacionih problema, kod diskretnih problema cilj je da se minmizuje ili maksimizuje zadata funkcija cilja, sa, ili bez ograničenja.

U grupu kombinatorne optimizacije spada veliki broj dobro poznatih problema optimizacije, poput problema trgovačkog putnika, transportnih problema, problema optimalnog rasporeda radne snage, izbora projekata, oročavanja novca u banci, raspoređivanja stvari u ranac, pakovanja proizvoda, kao i mnogi problemi iz vojnih disciplina, kao što su problem izbora borbenih sredstava i problem vojne strategije.

Opšta formulacija kombinatornog problema može da se prikaže kao: dat je konačan ili prebrojivo beskonačan skup S i funkcija oblika $f : S \to R$. Potrebno je naći minimum funkcije f na skupu S:

$$\min \ f(x), \ x \in S, \tag{1.2}$$

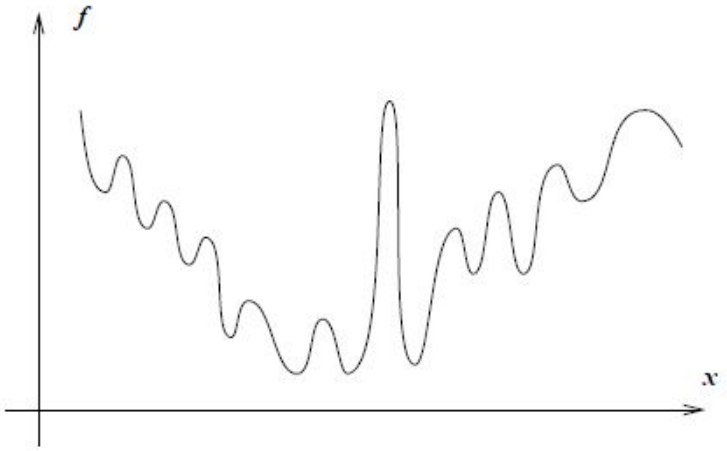
gde se S naziva dopustivi skup, a f je funkcija cilja. Potrebno je naći sva, ili bar neka dopustiva rešenja x^*, takva da je zadovoljena jendakost $f(x^*) = \min \{f(x) \mid x \in S\}$.

Među važnijim instancama kombinatornih problema su problemi linearnog celobrojnog programiranja. Mnogi problemi celobrojne optimizacije kao matematički model imaju graf. Ova klasa problema obuhvata probleme poput pronalaženja najkraće putanje između dva čvora (eng. shortest path problem - SPP), pronalaženje mini-

malnog povezujućeg (razapinjućeg) stabla (eng. minimum spanning tree problem - MSTP) i minimalnog pokrivača čvorova - dominirajućeg skupa (eng. minimum vertex cover problem - MVCP). Mnogi praktični problemi, kao što su optimizacija telekomunikacionih mreža ili optimizacija sistema navigacije svode se na ovakve probleme koji se modeliraju grafovima.

1.1.1.2 Globalna optimizacija

Globalna optimizacija se odnosi na skup problema optimizacije kod kojih varijable uzimaju realne vrednosti i kod kojih je potrebno pronaći globalno najbolje rešenje. Suštinski problem globalne optimizacije je mogućnost postojanja jako velikog broja lokalnih optimuma, koje je nemoguće sve pretražiti. Primer funkcije sa više lokalnih minimuma dat je na Slici 1.1.



Slika 1.1: Primer problema globalne optimizacije [3]

Neke metode optimizacije traže samo lokalno rešenje, tj. tačku koja ima manju vrednost funkcije cilja od tačaka u njenom susedstvu [3]. Kao primeri takvih metoda navode se Njutnov metod, ili metod silaska nizbrdo. Ovakve metode, očigledno, su neprimenljive za probleme globalne optimizacije.

Problem globalne neprekidne optimizacije može da se prikaže na sledeći način:

$$\min \ f(x), \tag{1.3}$$

gde je $x \in R^n$ realni vektor sa $n \geq 1$ komponentom, a funkcija koja se minimizuje

ima oblik: $f : R^n \rightarrow R$. Obično se znanje o funkciji f svodi na mogućnost da se izračuna njena vrednost u proizvoljnoj tački.

Tačka x^*, $x \in R^n$ je globalni minimum ako važi da je $\forall x,\ f(x^*) \leq f(x)$. Pronalaženje globalnog minimuma je teško, pošto je znanje o funkciji f samo lokalnog karaktera.

1.1.2 Podela prema ograničenjima

Prema kriterijumu postojanja ograničenja, problemi optimizacije mogu biti sa ili bez ograničenja (ovo se odnosi i na probleme kombinatorne i na probleme globalne optimizacije). Ograničenja značajno sužavaju dopustivi region prostora pretrage, pa su takvi problemi teži.

1.1.2.1 Optimizacija bez ograničenja

Optimizacija bez ograničenja (eng. unconstrained optimization) se još naziva i optimizacija sa ograničenjima vrednosti varijabli (eng. bound-constrained optimization), zato što kod ove klase problema varijable uzimaju vrednosti u okviru dozvoljenih donjih i gornjih granica.

Problemi globalne neprekidne optimizacije bez ograničenja (eng. continious global unconstrained optimization) mogu da se formulišu na sledeći način:

$$\min\ f(x),\ x = (x_1, x_2, ..., x_n) \in S \subseteq R^n, \tag{1.4}$$

gde je $S \subseteq R^n$ prostor pretrage, a n dimenzija problema. S je n-dimenzioni hiperkvadar u prostoru R^n koji se definiše pomoću gornjih i donjih granica vrednosti parametara:

$$lb_i \leq x_i \leq ub_i,\ 1 \leq i \leq n, \tag{1.5}$$

gde su lb_i i ub_i donja i gornje granica i-tog parametra, respektivno.

1.1.2.1.1 Primeri benčmark funkcija bez ograničenja

Najpoznatiji benčmark primer kombinatorne optimizacije je problem trgovačkog putnika. Dat je skup od n gradova koje trgovački putnik treba da poseti tačno po jedanput takvim redosledom da troškovi puta budu minimalni. Dopustivi skup S čine sve permutacije skupa $\{1, ..., n\}$, a ima ih $n!$. Za datu permutaciju $p = j_1, ..., j_n$ funkcija cilja $f(p)$ je zbir troškova putovanja između gradova j_1 i j_2, j_2 i j_3,..., j_{n-1} i j_n.

Benčmark problemi globalne neprekidne optimizacije bez ograničenja imaju veliki broj lokalnih optimuma. Za testiranje metoda optimizacije, u literaturi se najčešće koriste problemi sa 5, 10, 30 i 50 varijabli, a ako varijabli ima 100, 500, 1000, ili više, onda se takvi primeri svrstavaju u grupu problema velikog obima (eng. large-scale optimization).

U nastavku su date formulacije nekih od najpopularnijih benčmark problema za globalnu neprekidnu optimizaciju bez ograničenja. U prikazanim formulacijama, D je broj varijabli problema.

Funkcija *Sphere* ili *De Jong* (neprekidna, diferencijabilna, skalabilna, multimodalna):

$$f(x) = \sum_{i=1}^{D} x_i^2, \tag{1.6}$$

uz ograničenje vrednosti varijabli $0 \leq x_i \leq 10$. Globalni optimum se nalazi u tački $x^* = f(0, ..., 0),\ f(x^*) = 0$.

Sphere je funkcija koja se obično koristi kao prvi primer na kome se vrši testiranje optimizacionog metoda. Iako je ova funkcija dobro poznata, proces optimizacije predpostavlja da ne postoji nikakvo znanje o njenom analitičkom obliku, izvodima, itd. Jedino može da se izračuna njena vrednost u bilo kojoj tački.

Funkcija *Rastrigin* (neprekidna, diferencijabilna, skalabilna, multimodalna):

$$f(x) = 10D + \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i)), \tag{1.7}$$

uz ograničenje vrednosti varijabli $-5.12 \leq x_i \leq 5.12$. Globalni optimum se nalazi u

tački $x^* = f(0, ..., 0),\ f(x^*) = 0.$

Rastrigin funkcija ima oblik sfere sa dodatim malim kosinusnim talasima i obično se koristi kao drugi test problem. Jednostavnije metode optimizacije koje "srljaju" ka lokalnom minimumu će u slučaju Rastrigin-a da zaglave.

Nakon Sphere i Rastrigin, obično se koriste još komplikovanije benčmark funkcije.

Funkcija *Ackley* (neprekidna, diferencijabilna, skalabilna, multimodalna):

$$f(x) = -200e^{-0.02\sqrt{x_1^2+x_2^2}}, \tag{1.8}$$

uz ograničenje vrednosti varijabli $-32 \leq x_i \leq 32$. Globalni optimum se nalazi u tački $x^* = (0, 0), f(x^*) = -200.$

Funkcija *Schwefel* (neprekidna, diferencijabilna, skalabilna, unimodalna):

$$f(x) = D * 418.9829 + \sum_{i=1}^{D} -x_i \sin(\sqrt{|x_i|}), \tag{1.9}$$

uz ograničenje vrednosti varijabli $-500 \leq x_i \leq 500$. Globalni optimum se nalazi u tački $x^* = f(420.9867, ..., 420.9867),\ f(x^*) = 0.$

Funkcija *Step* (prekidna, nediferencijabilna, skalabilna, unimodalna):

$$f(x) = \sum_{i=1}^{D}(|x_i|), \tag{1.10}$$

uz ograničenje vrednosti varijabli $-100 \leq x_i \leq 100$. Globalni optimum se nalazi u tački $x^* = f(0, ..., 0), f(x^*) = 0.$

1.1.2.2 Optimizacija sa ograničenjima

Ako se u formulaciju optimizacionog problema uključe ograničenja, dopustivi region prostora pretrage se značajno sužava, pa i sam proces optimizacije postaje teži. Za rešenja koja zadovoljavaju ograničenja kaže se da su dopustiva (eng. feasible), dok se rešenja koja ne zadovoljavaju ograničenja nazivaju nedopustiva (eng. infeasible).

Problemi kombinatorne optimizacije sa ograničenjima (eng. combinatorial constrained optimization) definišu se kao skup od tri elementa (S, f, Ω), gde S označava prostor pretrage, f je funkcija cilja koju je potrebno minimizovati ili maksimizovati, dok Ω označava skup ograničenja koja moraju da se zadovolje kako bi dobijena rešenja bila dopustiva [4]. Cilj je da se pronađe globalno optimalno rešenje s^*, koje u slučaju problema minimizacije ima najmanju vrednost funkcije cilja, a da pritom zadovoljava sva ograničenja data u formulaciji problema.

Globalni problemi sa ograničenjima (eng. continuous constrained optimization) mogu da se modeliraju pomoću sledećih jednačina:

$$\min \text{ (ili } \max) \ f(x), \ x = (x_1, x_2, ..., x_n), \tag{1.11}$$

gde je $x \in F \subseteq S$, a $S \subseteq R^n$, gde je S hiper-kvadar određen jednačinom (1.5), a domen prostora pretrage sa dopustivim rešenjima $F \subseteq S$ definiše se pomoću skupa od m linearnih i nelinearnih ograničenja:

$$\begin{aligned} g_j(x) \leq 0, \ \text{za } j = 1, ..., q \\ h_j(x) = 0, \ \text{za } j = q+1, ..., m, \end{aligned} \tag{1.12}$$

gde q predstavlja broj ograničenja sa nejadnakosti (eng. inequality constraints), dok je $(m - q)$ broj ograničenja sa jednakostima (eng. equality constraints). Pritom, u svakoj tački $x \in F$ sva ograničenja g_k koja zadovoljavaju jednakost $g_k(x) = 0$ nazivaju se aktivna ograničenja u tački x.

Tačka $x^* \in S$ koja zadovoljava sva ograničenja i u slučaju problema minimizacije uslov da je:

$$f(x^*) \leq f(x), \ \forall x \in S \tag{1.13}$$

naziva se globalno optimalno rešenje problema.

1.1.2.2.1 Upravljanje ograničenjima

U literaturi je poznat veliki broj metoda za upravljanje ograničenjima. Prema jednoj uprošćenoj taksonomiji koja se bazira na taksonomijama predloženim u [5] i [6], tehnike za upravljanje ograničenjima mogu da se podele u sledeće grupe [7]:

- kaznene funkcije;
- dekoderi;
- posebni operator i
- odvajanje funkcije cilja i ograničenja.

Uopštena formulacija kaznenih funkcija (eng. penalty function) može da se prikaže kao [7]:

$$\phi(\vec{x}) = f(\vec{x}) + p(\vec{x}), \tag{1.14}$$

gde $\phi(\vec{x})$ označava proširenu funkciju cilja, dok je $p(\vec{x})$ kaznena vrednost koja se izračunava kao [7]:

$$p(\vec{x}) = \sum_{i=1}^{m} r_i * \max(0, g_i(\vec{x}))^2 + \sum_{j=1}^{p} c_j * |h_j(\vec{x})|, \tag{1.15}$$

gde su r_i i c_j pozitivne konstante koje se nazivaju kazneni faktori.

Najjednostavnija kaznena funkcija je u literaturi poznata kao "smrtna kazna" koja radi tako što najlošijim rešenjima dodeljuje najgoru vrednost, koja se na taj način eliminišu iz optimizacionog procesa. Postoji veliki broj kaznenih funkcija koje svoje kaznene faktore drže fiksirane tokom celog optimizacionog procesa. Glavni nedostatak ovih pristupa je generalizacija, npr. vrednosti koje su odgovarajuće za jedan problem, često nisu za drugi. S druge strane, postoji takođe i veliki broj tzv. adaptivnih kaznenih funkcija, koje koriste informacije iz optimizacionog procesa i na osnovu njih dinamički ažuriraju svoje kaznene faktore.

Uprkos jednostavnoj implementaciji, primena kaznenih funkcija zahteva pažljivo podešavanje vrednosti kaznenih faktora koji određuju "ozbiljnost" kazni koje se primenjuju na nedopustiva rešenja. Univerzalna podešavanja ne postoje, već je potrebno

za svaki problem pronaći optimalnu vrednost kaznenih faktora [8].

Dekoderi (eng. decoders) su dugo vremena smatrani za najbolju tehniku za upravljanje ograničenjima. Oni se oslanjaju na ideju mapiranja dopustivog regiona F prostora pretrage S na dekodirani prostor koji je lakši za optimizaciju. Proces mapiranja dekodera mora da garantuje da je svako dopustivo rešenje iz S uključeno u dekodirani prostor i da svakom dekodiranom rešenju odgovara određeno dopustivo rešenje u S. Osim toga, proces transformacije mora da bude brz i poželjno je da male promene u S originalnog problema impliciraju male promene dekodiranog prostora [7]. Koziel i Michalewicz su predložili HM (homomorphous maps), gde se dopustivi region mapira na n-dimenzionu kocku [9]. Iako su dekoderi interesantna tehnika iz teoretske perspektive, prilično je komplikovano da se implementiraju i značajno povećavaju troškove izračunavanja, pa se zbog toga danas retko koriste.

Posebni operator (eng. special operator) upotrebljava se dvojako, ili kao način da se očuva dopustivost rešenja, ili kao način za preusmeravanje procesa pretrage u okviru interesantnog dela prostora pretrage, kao što su na primer granice dopustivog domena. Michalewicz je predložio operator GENOCOP koji radi samo sa linearnim ograničenjima i koji zahteva da početno rešenje bude dopustivo i proces predobrade koji eliminiše ograničenja jednakosti i neke od varijabli optimizacionog problema.

Za razliku od kaznenih funkcija koje kombinuju funkciju cilja i ograničenja u jednu funkciju, tehnika odvajanja funkcije cilja od ograničenja radi potpuno suprotno. Tako na primer, po jednom pristupu, kvalitet rešenja $\vec{x}$ se određuje prema sledećoj jednačini [7]:

$$kvalitet(\vec{x}) = \begin{cases} f(\vec{x}), \text{ ako je dopustivo} \\ 1 + r(\sum_{i=1}^{m} g_i(\vec{x}) + \sum_{j=1}^{p} h_j(\vec{x})), \text{ ako je nedopustivo} \end{cases} \tag{1.16}$$

Iz jednačine (1.16) se vidi da je kvalitet dopustivog rešenja uvek bolji od kvaliteta nedopustivog rešenja, čiji kvalitet zavisi od sume vrednosti prekoračenja ograničenja.

Od savremenih pristupa za upravljanje ograničenjima potrebno je posebno izdvoljiti metodu koju je predložio Deb [10]. Ova tehnika, koja je u literaturi poznata pod

nazivom Debova pravila, smatra se za jednu od najefektivnijih metoda za upravljanje ograničenjima. Ovaj pristup koristi skup od tri kriterijuma dopustivosti:

1. kada se porede dva dopustiva rešenja, bira se ono koju ima bolju vrednost funkcije cilja;
2. kada se poredi jedno dopustivo sa jednim nedopustivim rešenjem, uvek se bira dopustivo rešenje i
3. kada se porede dva nedopustiva rešenja, bira se ono koje ima manju vrednost sume prekoračenja ograničenja.

Suma vrednosti prekoračenja ograničenja računa se na sledeći način [10]:

$$\phi(\vec{x}) = \sum_{i=1}^{m} \max(0, g_i(\vec{x}))^2 + \sum_{j=1}^{p} |h_j(\vec{x})|, \tag{1.17}$$

gde su vrednosti svakog ograničenja nejednakosti $g_i(\vec{x}), i = 1, ..., m$ i ograničenja jednakosti $h_j(\vec{x}), j = 1, ..., p$ normalizovane.

Stohastičko rangiranje (SR), koje je kreirano sa ciljem da se isprave nedostaci kaznenih funkcija (prevelike ili premale kazne zbog pogrešnih vrednosti kaznenih faktora), su prvi predložili Runarsson i Yao [8]. Umesto kaznenih faktora, stohastičko rangiranje implementira parametar P_f koji kontroliše kriterijume za poređenje nedopustivih rešenja i to: na osnovu sume vrednosti prekoračenja ograničenja ili samo na osnovu vrednosti funkcije cilja.

Od novijih tehnika korisno je još spomenuti i ε metod za upravljanje ograničenjima, koji su predložili Takahama i Sakai [11]. Ovaj mehanizam transformiše problem sa ograničenjima u problem bez ograničenja i sastoji se iz dve osnovne komponente: relaksacija ograničenja jednakosti time što se oko hiper-ravni koju definiše to ograničenje uvodi pojas širine ε, unutar koga se rešenja smatraju dopustivim i dinamičko sužavanje toga pojasa da bi na kraju dobijena rešenja bila dopustiva.

Osim pomenutih, u literaturi se još pronalaze tehnika pravila dopustivosti (eng. feasibility rules) i metode savremenih kaznenih funkcija i posebnih operatora [7].

1.1.2.2.2 Primeri benčmark funkcija sa ograničenjima

Kao najčešći skup benčmark funkcija za globalnu optimizaciju sa ograničenjima, koje se koriste u svetskoj literaturi, navode se dobro poznate G funkcije ($G1 - G24$). Ove funkcije su prvi put definisane za potrebe svetskog kongresa $IEEE\ CEC$ 2006 [12]. Ovde se kao primer navode prve dve.

$G1$:

$$\min\ f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{13} x_i \tag{1.18}$$

uz ograničenja:

$$\begin{aligned} g_1(\vec{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\ g_2(\vec{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\ g_3(\vec{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\ g_4(\vec{x}) &= -8x_1 + x_{10} \leq 0 \\ g_5(\vec{x}) &= -8x_2 + x_{11} \leq 0 \\ g_6(\vec{x}) &= -8x_3 + x_{12} \leq 0 \\ g_7(\vec{x}) &= -2x_4 - x_5 + x_{10} \leq 0 \\ g_8(\vec{x}) &= -2x_6 - x_7 + x_{11} \leq 0 \\ g_9(\vec{x}) &= -2x_8 - x_9 + x_{12} \leq 0, \end{aligned} \tag{1.19}$$

gde su granice $0 \leq x_i \leq 1$ $(i = 1, ..., 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$ i $0 \leq x_{13} \leq 1$.

Globalni minimum nalazi se u tački $\vec{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ sa šest aktivnih ograničenja $(g_1, g_2, g_3, g_7, g_8, g_9)$ i $f(\vec{x}^*) = -15$.

$G2$:

$$f(\vec{x}) = -|\frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}| \tag{1.20}$$

uz ograničenja:

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \leq 0 \tag{1.21}$$
$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0,$$

gde je $n = 20$ i $0 < x_i \leq 10$ $(i = 1, ..., n)$.

Za ovu funkciju nije poznat globalni optimum, a najbolje poznato rešenje nalazi se u tački $\vec{x}^* = (3.16246061572185, 3.12833142812967, 3.09479212988791,$ $3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285,$ $2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865,$ $0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199,$ $0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760,$ $0.44038285956317)$, dok najbolja prijavljena funkcija cilja ima vrednost $f(\vec{x}^*) = -0.80361910412559$.

U Tabeli 1.1, prikazane su osobine 24 benčmark funkcije ($G1$-$G24$). U prikazanoj tabeli, n označava broj varijabli problema, ρ je odnos između dopustivog regiona i ukupnog prostora pretrage (F/S), LN and NN su brojevi linearnih i nelinearnih ograničenja nejednakosti, respektivno, LJ i NJ su brojevi linearnih i nelinearnih ograničenja jednakosti, respektivno, dok je a broj aktivnih ograničenja u rešenju $\vec{x}$ [12].

Tabela 1.1: Karakteristike G1-G24 benčmark funkcija

prob.	**n**	**vrsta funkcije**	ρ	**LN**	**NN**	**LJ**	**NJ**	**a**
g01	13	kvadratna	0.0111%	9	0	0	0	6
g02	20	nelinearna	99.9971%	0	2	0	0	1
g03	10	polinomna	0.0000%	0	0	0	1	1
g04	5	kvadratna	52.1230%	0	6	0	0	2
g05	4	kubna	0.0000%	2	0	0	3	3
g06	2	kubna	0.0066%	0	2	0	0	2
g07	10	kvadratna	0.0003%	3	5	0	0	6
g08	2	nelinearna	0.8560%	0	2	0	0	0
g09	7	polinomna	0.5121%	0	4	0	0	2
g10	8	linearna	0.0010%	3	3	0	0	6
g11	2	kvadatna	0.0000%	0	0	0	1	1
g12	3	kvadratna	4.7713%	0	1	0	0	0
g13	5	nelinearna	0.0000%	0	0	0	3	3
g14	10	nelinearna	0.0000%	0	0	3	0	3
g15	3	kvadratna	0.0000%	0	0	1	1	2
g16	5	nelinearna	0.0204%	4	34	0	0	4
g17	6	nelinearna	0.0000%	0	0	0	4	4
g18	9	kvadratna	0.0000%	0	13	0	0	6
g19	15	nelinearna	33.4761%	0	5	0	0	0
g20	24	linearna	0.0000%	0	6	2	12	16
g21	7	linearna	0.0000%	0	1	0	5	6
g22	22	linearna	0.0000%	0	1	8	11	19
g23	9	linearna	0.0000%	0	2	3	1	6
g24	2	linearna	79.6556%	0	2	0	0	2

Osim G funkcija, u literaturi se koristi i veliki broj benčmark funkcija koje liče na uobičajene inženjerske probleme. Jedan od najpopularnijih primera je problem zavarivanja grede (eng. welded beam design). Cilj je da se iskonstruiše zavarena greda uz minimalne troškove i određena ograničenja. Minimizacija troškova vrši se pomoću četiri varijable x_1, x_2, x_3 i x_4 i uz ograničenja pritiska sečenja (ς), pritiska savijanja grede (σ), tereta deformisanja grede (P_c) i savijanja grede (δ).

Problem zavarivanja grede se formuliše na sledeći način:

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{1.22}$$

uz ograničenja:

$$\begin{aligned} g_1(\vec{x}) &= \varsigma(\vec{x}) - 13,600 \leq 0 \\ g_2(\vec{x}) &= \sigma(\vec{x}) - 30,000 \leq 0 \\ g_3(\vec{x}) &= x_1 - x_4 \leq 0 \\ g_4(\vec{x}) &= 0.10471(x_1^2) + 0.04811x_3x_4(14 + x_2) - 5.0 \leq 0 \\ g_5(\vec{x}) &= 0.125 - x_1 \leq 0 \\ g_6(\vec{x}) &= \delta(\vec{x}) - 0.25 \leq 0 \\ g_7(\vec{x}) &= 6,000 - P_c(\vec{x}) \leq 0 \end{aligned} \tag{1.23}$$

gde je:

$$\begin{aligned} \varsigma(\vec{x}) &= \sqrt{\varsigma'^2 + (2\varsigma'\varsigma'')\frac{x^2}{2R} + \varsigma''^2} \\ \varsigma' &= \frac{6,000}{\sqrt{2x_1}x_2} \\ \varsigma'' &= \frac{MR}{J} \\ M &= 6,000(14 + \frac{x_2}{2}) \\ R &= \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2} \\ J &= 2\{x_1x_2\sqrt{2}[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2]\} \\ \sigma(\vec{x}) &= \frac{504,000}{x_4x_3^2} \\ \delta(\vec{x}) &= \frac{65,865,000}{(30x10^6)x_4x_3^x} \\ P_c(\vec{x}) &= \frac{4.013(30x10^6)\sqrt{\frac{x_3^2x_4^6}{36}}}{196}(1 - \frac{x_3\sqrt{\frac{30x10^6}{4(12x10^6)}}}{28}), \end{aligned} \tag{1.24}$$

gde su granice varijabli $0.1 \leq x_1, x_4 \leq 2.0$ i $0.1 \leq x_2, x_3 \leq 10.0$. Globalni minimum se nalazi u tački $\vec{x}^* = (0.205730, 3.470489, 9.036624, 0.205729)$ i $f(\vec{x}^*) = 1.724852$.

1.1.3 Podela prema broju funkcija cilja

Na osnovu broja funkcija cilja, problemi optimizacije mogu biti sa jednom funkcijom cilja, ili sa više funkcija cilja. Svi optimizacioni problemi o kojima je do sada bilo reči pripadaju grupi problema sa jednom funkcijom cilja (eng. single-objective optimization).

1.1.3.1 Optimizacija sa više funkcija cilja

Problemi sa više funkcija cilja (eng. multi-objective optimization - MOO) definišu se pomoću više funkcija cilja koje je potrebno simultano optimizovati. Ovaj problem je u literaturi poznat i kao više-ciljno programiranje i višekriterijumska optimizacija.

Ovi optimizacioni problemi imaju široku primenu u mnogim oblastima, kao što su građevinarstvo, arhitektura, šumarstvo, itd., gde se donosi optimalna odluka kao rezultat kompromisa između ciljeva koji su najčešće u suprotnosti jedni sa drugim. Kao primer može da se navede problem koji se javlja prilikom dizajniranja računarske mreže, kada je potrebno maksimizovati performanse, pouzdanost i sigurnost mreže uz minimizaciju troškova harverskih komponenti, itd.

Jedan najčešćih i najšire korišćenih načina u literaturi za rešavanje MOO problema jeste pristup težinske sume (eng. weighted sum). Ovaj metod agregira sve ciljeve u jedan tako što množi vrednost svakog cilja odgovarajućim težinskim koeficijentom. Pitanje koje se pritom postavlja jeste koja vrednost težinskog koeficijenta da se dodeli svakom cilju. U slučaju kada dve funkcije cilja imaju vrlo različite numeričke vrednosti (na primer jedna u intervalu $[0, 1]$, a druga u opsegu $[100, 200]$), onda se vrši skaliranje.

Za svaki cilj $f_1(x), ..., f_n(x)$ bira se skalarni težinski koeficijent ω, tako da se rešavanje problema svodi na optimizaciju složene funkcije cilja U [13]:

$$U = \sum_{i=1}^{n} \omega_i f_i(x) \tag{1.25}$$

Kao još jedan način za rešavanje MOO problema navodi se tehnika hijerarhije ciljeva. Prema ovog pristupu, svakom cilju se određuju prioriteti. Prvo se optimizuje najvažniji cilj (sa najvećim prioritetom), i tek kada se on optimizuje, pristupa se optimizaciji drugog, trećeg, itd. cilja. Ovaj pristup ima veliki broj praktičnih primena u literaturi [14].

Ipak, za najprecizniju MOO optimizaciju koristi se koncept Pareto optimalnosti. MOO problem matematički može da se formuliše kao [15]:

$$\min\ [f_1(x), f_2(x), ..., f_n(x)],\ x \in S, \tag{1.26}$$

gde je f skalarna funkcija, $n > 1$, a S je skup ograničenja određen jednačinom:

$$S = \{x \in R^m : h(x) = 0,\ g(x) \geq 0\} \tag{1.27}$$

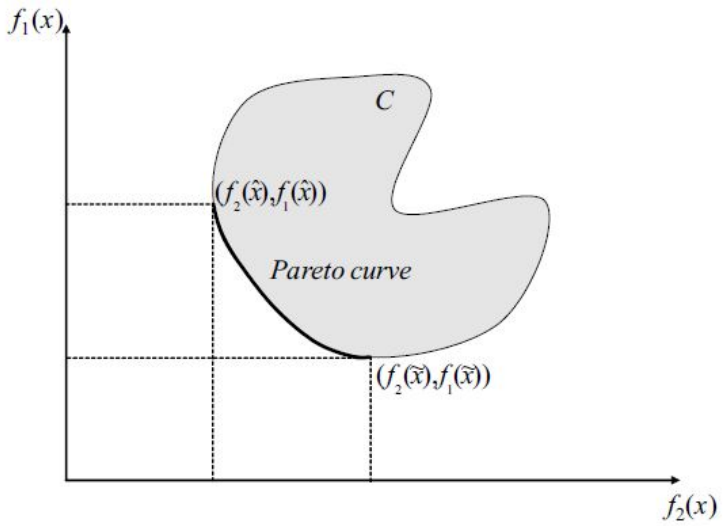
Skalarni koncept "optimalnosti" ne može direktno da se primeni na MOO, pa je potrebno da se definiše Pareto optimalnost. Ukratko, za vektor $x^* \in S$ kaže se da je Pareto optimalan ako svi drugi vektori $x \in S$ imaju veću vrednost za makar jednu funkciju cilja $f_i, i = 1, ..., n$, ili imaju istu vrednost za sve funkcije ciljeva kao i rešenje koje je proglašeno da je Pareto optimalno.

Za tačku x^* kaže se da je slabi Pareto optimum ili slabo efikasno rešenje MOO problema ako i samo ako $\nexists x \in S$ takvo da je $f_i(x) < f_i(x^*)$, za $\forall i \in \{1, ..., n\}$. S druge strane, tačka x^* je jak Pareto optimum ili jako efikasno rešenje MOO problema ako i samo ako $\nexists x \in S$ takvo da je $f_i(x) \leq f_i(x^*)$, za $\forall i \in \{1, ..., n\}$, sa makar jednom striktnom nejednakosti.

Osim slabog i jakog Pareto optimuma, postoje takođe i lokalna Pareto optimalna rešenja, za koje važi ista definicija, jedino što se posmatraju samo dopustiva rešenja u susedstvu tačke x^*. Drugim rečima, ako je $B(x^*, \epsilon)$ lopta prečnika $\epsilon > 0$ oko tačke x^*, potrebno je da za neko $\epsilon > 0$ bude zadovoljeno da $\nexists x \in S \cap B(x^*, \epsilon)$, takvo da

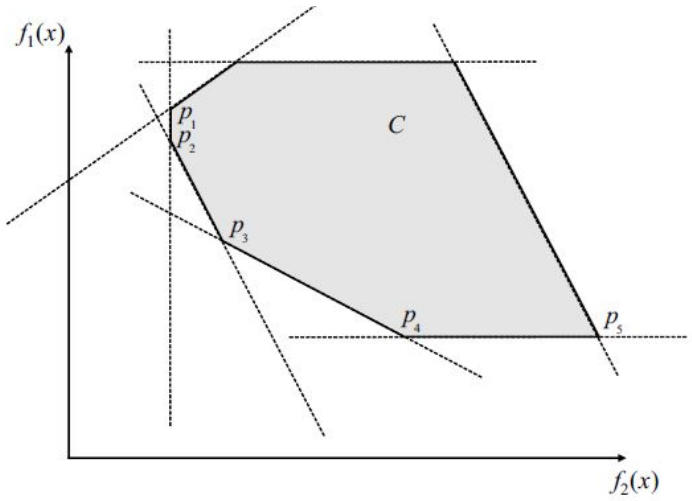
je $f_i(x) \leq f_i(x^*)$, za $\forall i \in \{1, ..., n\}$, sa makar jednom striktom nejednakosti.

Skup svih efikasnih rešenja naziva se Pareto front (kriva, površina). Oblik Pareto fronta odslikava kompromis u optimizaciji funkcija ciljeva koje su u suprotnosti jedna sa drugom. Primer Pareto krive je prikazan na Slici 1.2, gde sve tačke između $(f_2\tilde{x}, f_1\tilde{x})$ i $(f_2\hat{x}, f_1\hat{x})$ definišu Pareto front. Ove tačke se nazivaju nedominirajuća rešenja.



Slika 1.2: Primer Pareto krive [15]

Primer jakog i slabog Pareto optimuma prikazan je na Slici 1.3. Tačke p_1 i p_5 su slabi Pareto optimumi, dok su tačke p_2, p_3 i p_4 jaki Pareto optimumi.



Slika 1.3: Primer jakog i slabog Pareto optimuma [15]

1.1.3.1.1 Primeri benčmark funkcija sa više kriterijuma

U cilju testiranja algoritama za MOO probleme najčešće se koristi familija ZDT funkcija. U nastavku je dat primer test funkcije.

$ZDT2$: Broj dimenzija ovog problema je $n = 30$ sa nekonveksnim Pareto optimalnim frontom:

$$ZDT2 = \begin{cases} \min \ f_1(x) = x_1 \\ \min \ f_2(x) = g(x)[1 - (x_1/g(x)^2)] \\ g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1), \end{cases} \tag{1.28}$$

gde sve varijable imaju vrednosti iz opsega $[0, 1]$. Pareto optimalni region je $0 \leq x_i^* \leq 1$ i $x_1^* = 0$ za $i = 2, 3, ...30$.

1.2 Složenost algoritama i klase kompleksnosti

Često se dešava da računarski program, tj. algoritam radi na instancama problema sa manjim brojem ulaza, ali izvršavanje na instancama sa većim brojem ulaza traje neprihvatljivo dugo. Zbog toga je potrebno uvesti neku kvantifikaciju vremena izvršavanja algoritama.

Važan aspekt svakog algoritma je pored vremena izvršavanja (vremenska složenost), takođe i potrošnja memorijskih resursa (prostorna složenost). Evaluacija vremenske složenosti algoritma sastoji se u broju računarskih koraka koje treba izvršiti. Međutim, računarski korak podrazumeva različite operacije, kao što su na primer sabiranje i deljenje, čije izvršavanje traje različito. Osim toga, vreme izvršavanja zavisi i od konkretne računarske platforme na kojoj se algoritam izvršava. Zato se u algoritmu definiše osnovni korak koji traje jednu jedinicu i u odnosu na njega se vrši evaluacija.

Prostorna složenost algoritma podrazumeva veličinu memorije koja je potrebna za izvršavanje algoritma, pri čemu se prostor za skladištenje ulaznih podataka obično zanemaruje. Ovo omogućava upoređivanje različitih algoritama za rešavanje istog

problema.

S obzirom na to da se svaki optimizacioni problem sastoji iz skupa zadataka, problem je algoritamski rešiv ako postoji algoritam koji može uspešno da se primeni na svaki individualni zadatak problema koji se optimizuje. Individualni zadatak predstavlja instancu datog problema sa konkretnim parametrima. Međutim, osim potrebe da postoji algoritam, algoritam treba da bude i efektivan tako što će zadatak moći da reši u razumnom vremenu uz raspoložive resurse.

Teorija kompleksnosti izračunavanja (eng. computational complexity theory) omogućava kategorizaciju problema prema njihovoj težini. Težina problema se definiše na osnovu količine minimalnih računarskih resursa (bilo prostornih, bilo vremenskih) koji su neophodni za rešavanje problema [16]. Zbog ovoga je težina problema u tesnoj korelaciji sa složenošću algoritma. Na osnovu prostorne i vremenske složenosti algoritma, mogu da se definišu donja i gornja granica težine problema [2].

Problem pripada klasi složenosti P (eng. polynomial time), ako može da se reši, u opštem slučaju, nekim od poznatih algoritama polinomske složenosti. Klasa problema NP (eng. non-deterministic polynomial time) opisuje skup problema za koje važi da se rešenje datog problema može verifikovati algoritmom polinomske složenosti.

NP-teški problemi su problemi za koje ne postoji polinomski algoritam i koji mogu da se transformišu jedan u drugi u polinomskom vremenu. Drugim rečima, algoritam koji se koristi za rešavanje NP-teškog problema može se u polinomskom vremenu redukovati na drugi algoritam koji može da reši bilo koji problem iz klase NP [2]. Problem je NP-kompletan ako pripada klasi NP i ako svi ostali NP problemi mogu da se algoritmom polinomske složenosti svedu na dati problem.

Detaljnija analiza složenosti data je u [2], [16] i [17].

1.3 Taksonomija optimizacionih algoritama

Postoji više kriterijuma za klasifikaciju optimizacionih algoritama. Jedna od podela algoritama je na determinističke (eng. deterministic) i stohastičke (eng. stohastic).

Deterministički algoritam je algoritam koji pri svakom izvršavanju, pod bilo kojim uslovima, od istog ulaza doći do istog izlaza, sledeći svaki put istu putanju (isti niz naredbi). Međutim, za probleme sa više dimenzija i komplikovanije funkcije koje uključuju više lokalnih optimuma, deterministički algoritmi nisu u stanju da pronađu optimalno rešenje, pošto je za njegovo pronalaženje potrebno mnogo računarskih resursa. U slučaju NP-teških problema, koliko god da je veliko povećanje snage, primenom determinističkih metoda, neće biti dovoljno da se vreme izračunavanja smanji na prihvatljivo.

Neki deterministički algoritmi izvršavaju se u iteracijama. Izvršavanje algoritma A je iterativni proces ako je cilj da se generiše novo i bolje rešenje x^{t+1} na osnovu modifikacija trenutnog rešenja x^t u iteraciji, ili vremenskom trenutku t. Tako na primer, deterministički Newton-Raphson metod koji pronalazi optimalnu vrednost funkcije $f(x)$ je ekvivalentan pronalaženju kritičnih tačaka od $f'(x) = 0$ u d-dimenzionom prostoru [18]:

$$x^{t+1} = x^t - \frac{f'(x^t)}{f''(x^t)} = A(x^t) \tag{1.29}$$

Iz jednačine (1.29) očigledno je da brzina konvergencije može da bude spora u blizini optimalne tačke, gde važi $f'(x) \to 0$. U velikom broju praktičnih primena brzina konvergencije algoritama nije zadovoljavajuća. Jednostavan način za poboljšavanje brzine konvergencije je uvođenje parametra p u jednačinu (1.29) [18]:

$$x^{t+1} = x^t - p\frac{f'(x^t)}{f''(x^t)}, \;\; p = \frac{1}{1 - A'(x^*)}, \tag{1.30}$$

gde je x^* optimalno rešenje ili fiksna tačka u formuli iteracije. U opštem slučaju, jednačina (1.30) može da se napiše kao:

$$x^{t+1} = A(x^t, p) \tag{1.31}$$

Prve metode koje su korišćene u rešavanju problema globalne optimizacije su bile deterministčke i uglavnom su bile bazirane na principu zavadi pa vladaj (eng. divide-and-conquer). One su prvi put primenjivane kasnih 50-tih godina prošlog veka, sa

rastom upotrebe elektronskih računara za potrebe istraživanja. Za razliku od ranijih principa, tehnika zavadi pa vladaj koristi iterativnu prirodu optimizacionih metoda koje se koriste na računarima: smanjivanje složenosti teoretskih struktura i oslonjanje na intenzivna izračunavanja koja istražuju prostor pretrage. Dakle, ova metoda deli problem na više podproblema i time smanjuje kompleksnost. Tipičan predstavnik ovih tehnika je algoritam grananja i ograničavanja (eng. branch and bound - BB), u nastavku *BB*. Ovaj algoritam generiše podprobleme tako što se glavni problem dekomponuje na grane (npr. svaka varijabla problema može da bude jedna grana) i kao takav može uspešno da se primeni na probleme sa diskretnim entitetima.

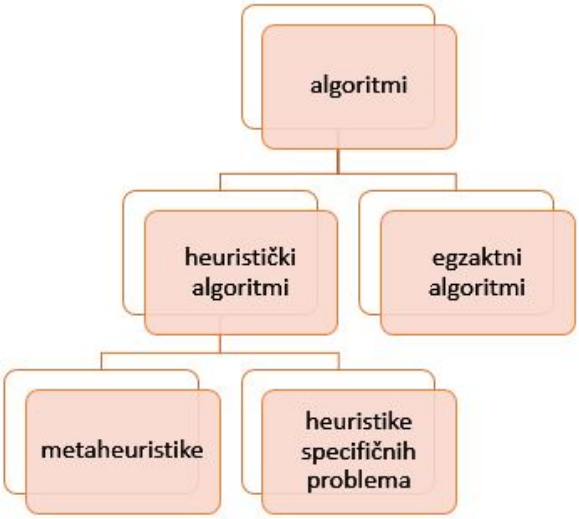
U literaturi se može naći veliki broj primena BB metoda. Tako na primer, BB je primenjen na širok spektar inženjerskih problema, kao što su problemi koji se odnose na proizvodne linije [19] i energetske sisteme [20]. Ovaj metod je takođe primenjivan i na teorijske grafovske probleme, kao što je pronalaženje najvećeg broja trouglova koji se ne dodiruju na neusmerenom grafu (eng. triangle packing problem - TP) [21].

S druge strane, stohastička optimizacija se odnosi na optimizaciju kada slučajni element utiče na izbor sledećeg koraka u izračunavanjima. Stohastički algoritam pri različitim startovanjima generalno ide različitim putanjama. Računari ne izvode slučajne izbore i kod njih se slučajnost simulira pomoću generatora pseudo-slučajnih sekvenci. U slučaju kada su potrebne prave slučajne sekvence, koriste se eksterni generatori slučajnosti bazirani na primer na prebrojavanju emitovanih čestica pri raspadanju nekog radioaktivnog elementa ili generatoru šuma. Dokazi za konvergenciju stohastičkih algoritama se izvode pomoću teorije verovatnoće. Dakle, ova grupa algoritama svoju pretragu bazira na slučajnosti.

Veliki broj optimizacionih algoritama i metoda pripada grupi stohastičkih optimizatora, kao što su Monte Karlo metod, evolutivni algoritmi, algoritmi inteligencije rojeva, itd. Sledeće poglavlje posvećeno je ovoj temi.

Prema još jednoj podeli, koja koristi malo drugačiju terminologiju od prethodne, optimizacioni algoritmi dele se na egzaktne i na heurističke [22]. Heuristički algoritmi

se dalje dele na heuristike specifičnih problema i na metaheuristike. Egzaktni algoritmi odgovaraju klasi determinističkih, a heuristički su iterativni koji od koraka do koraka korišćenjem trenutne situacije donose odluku kojim putem da nastave. Heurističke metode najčešće pronalaze zadovoljavajuća rešenja u relativno kratkom vremenskom periodu. Navedena podela je prikazana na Slici 1.4.



Slika 1.4: Podela optimizacionih algoritama

2 Metaheuristička optimizacija

2.1 Pojam i kategorizacija heuristika

Za razliku od egzaktnih metoda optimizacije koje garantuju pronalaženje optimalnog rešenja, heuristički algoritmi pokušavaju da pronađu što bolje rešenje, ali ne mogu da garantuju da je pronađeno rešenje i optimalno [23]. Za rešavanje mnogih praktičnih i benčmark problema nije moguće koristiti egzaktne metode zato što bi za njihovo rešavanje bilo potrebno mnogo vremena.

Heuristike se koriste za rešavanje konkretnih i specifičnih problema tako što koriste individualna svojstva samih problema pri njihovom rešavanju. Reč heuristika potiče od starogrčke reči *heuriskein*, što znači umetnost pronalaženja novog načina (pravila) u rešavanju problema. U poslednjim decenijama došlo je do naglog razvoja heurističkih metoda za rešavanje raznih optimizacionih problema. O ovome svedoči veliki broj objavljenih članaka i postojanje naučnih časopisa i konferencija koje su posvećene heuristikama.

Međutim, i pored osnovnog razloga korišćenja heuristika (nemogućnost egzaktnih metoda da pronađu optimalno rešenje u razumnom vremenskom periodu), postoje i dodatni razlozi, kao što su [23]:

- za dati problem ne postoji metoda koja može da ga reši na optimalni način;
- iako je raspoloživa egzaktna metoda za dati problem, ona ne može da se primeni na raspoloživom hardveru;
- heurističke metode su fleksibilnije od egzaktnih metoda, tako što omogućavaju modeliranje dodatnih karakteristika realnog problema i
- heuristička metoda se često koristi kao deo globalne procedure pretrage koja garantuje pronalaženje optimalnog rešenja datog problema.

Uopšteno je prihvaćeno da dobar heuristički algoritam treba da poseduje sledeća svojstva [23]: rešenje treba da se generiše u razumnom vremenskom periodu, heuristika treba da proizvede rešenje koje je blisko optimalnom sa velikom verovatnoćom i verovatnoća dobijanja lošeg rešenja bi trebala da bude minimalna.

S obzirom da se heurističke metode međusobno razlikuju, jako je teško da se da njihova potpuna klasifikacija [24]. Zbog toga se u literaturi nailazi na veliki broj taksonomija ovih metoda.

Prema jednoj podeli, heurističke metode dele se na [23]:

- dekompozicione metode (eng. decomposition methods) razlažu polazni problem na manje podprobleme koje je lakše rešiti, a pritom se vodi računa o činjenici da svi podproblemi pripadaju istoj glasi polaznog problema;
- induktivne metode (eng. inductive methods) generalizuju manje, ili jednostavnije verzije problema. Tehnike koje se mogu primeniti na ove verzije, mogu da se primene i na ceo problem;
- redukcione metode (eng. reduction methods) vrše identifikaciju atributa dobrih rešenja problema i uvode ih u optimizacioni proces u formi granica problema. Cilj je da se ograniči prostor potencijalnih rešenja simplifikacijom polaznog problema. Pritom se javlja rizik izostavljanja optimalnih rešenja početnog problema;
- konstruktivne metode (eng. constructive methods) grade rešenje od početka, po principu "korak po korak". U ovu grupu spadaju uglavnom deterministički algoritmi, koji se baziraju na najboljem izboru u svakoj iteraciji, i koriste se uglavnom za rešavanje standardnih problema kombinatorne optimizacije i
- metode lokalne pretrage (eng. local search methods) počinju proces pretrage tako što generišu dopustivo rešenje problema koje pokušavaju da poprave u iterativnom procesu. U svakom koraku ove procedure jedno rešenje se pomera ka drugom koje ima bolju vrednost funkcije cilja.

Osim navedenih metoda, značajne su i sledeće: heuristike matematičkog programiranja, kao i heuristike bazirane na podeli dopustivog skupa, restrikciji dopustivog skupa i na relaksaciji.

Heuristike matematičkog programiranja proces optimizacije počinju matematičkom formulacijom modela problema koji se optimizuje, a zatim pokušavaju da taj problem reše približnim, tj. aproksimativnim metodama. Cela oblast matematičkog programiranja može da se podeli na linearno, nelinearno, diskretno i stohastičko

programiranje i na teoriju igara. Zajednička osobina svih navedenih metoda je da one traže tačku u datom vektorskom prostoru koja zadovoljava neka (ili sva) ograničenja, a u kojoj funkcija koja se optimizuje dostiže ekstremnu vrednost.

Heuristike bazirane na dopustivom skupu prvo dele dopustivi skup na više podskupova, a zatim delimičnom pretragom svakog od podskupova pronalaze najbolje rešenje. Heurističko rešenje je ono kod kojega je funkcija cilja najbolja.

Metode utemeljene na restrikciji dopustivog skupa pretragu izvode tako što eliminišu određene podskupove iz dopustivog skupa restrikcijom, čime se prostor pretrage sužava. To omogućava da se novi zadatak lakše rešava.

Konačno, heuristike koje koriste relaksaciju funkcionišu po suprotnom principu od metoda restrikcija. Ove heuristike proširuju dopustivi skup, ali tako da se omogući jednostavnije rešavanje novog problema.

Od svih prikazanih heurističkih metoda, konstruktivne i metode lokalne pretrage često su primenljivane u metaheurističkim algoritmima.

2.2 Metaheuristički algoritmi

S obzirom na činjenicu da su se heurističke metode i algoritmi pokazali kao robusni optimizatori velikog broja problema, u poslednjoj deceniji su mnogi istraživači širom sveta pokazali veliko interesovanje za njihov razvoj i unapređivanje. Ovaj napor je rezultirao nastankom metaheuristika, kao univerzalnim heuristikama za širok spektar problema.

Nasuprotog heuristikama, metaheuristike obuhvataju generičke skupove pravila koja mogu da se primene na rešavanje velikog broja optimizacionih problema. Osnovni principi ovih algoritama baziraju se na opštim algoritmima optimizacije koji koriste iterativne mehanizme da bi popravili postojeće rešenje.

Od mnogobrojnih definicija metaheuristika, ovde se citira jedna po kojoj je metaheuristika skup algoritamskih koncepata koji se koriste za definisanje heurističkih metoda koje su primenljive na širok spektar problema [24]. Takođe, metaheuristika

može da se definiše i kao heuristika opšte namene čiji je glavni zadatak usmeravanje problemski specifičnih heuristika prema području prostora pretrage u kojem se nalaze dobra rešenja. Drugim rečima, metaheuristika je metod za pronalaženje dobre heuristike za određeni problem.

Koreni metaheurističkih metoda nalaze se u mašinskom učenju (eng. machine learning), evolutivnim algoritmima (eng. evolutionary algorithms - EA) i u fazi logici (eng. fuzzy logic). Metaheuristike daju odgovore na sledeća pitanja: koje vrednosti parametara daju dobre rezultate kada se primeni heuristika x na problem y; na koji način je moguće prilagoditi parametre heuristike x da bi se dobila bolja rešenja problema y i da li je bolja heuristika x ili heuristika y.

Uspešna primena metaheuristika u mnogim oblastima svedoči o njihovoj efikasnosti i efektivnosti u rešanju velikih i složenih problema. Neke od aplikacija metaheuristika za rešavanje svakodnevnih problema iz realnog okruženja obuhvataju [24]:

- problemi inženjerskog dizajna, optimizacija topologije i strukturalna optimizacija u elektronici i VLSI (very large scale integration) dizajnu, aerodinamici i dinamici fluida, telekomunikacijama, automobilskoj industriji i robotici;
- mašinskom učenju i iskopavanju podataka u bioinformatici, računarskoj biologiji i finansijama;
- sistemsko modeliranje, simulacija i identifikacija u fizici, hemiji i biologiji i
- problemi rutiranja i planiranja, problemi raspoređivanja i proizvodnje u logistici i transportu, problemi u upravljanju lancima snabdevanja, itd.

U literaturi postoji veliki broj kriterijuma za klasifikaciju metaheuristika. Prema jednoj podeli, metaheuristike se dele na [24]:

- determinističke (eng. deterministic) i stohastičke (eng. stochastic) metaheuristike. Determinističke metaheuristike pristupaju rešavanju problema optimizacije tako što donose determinističke odluke (primeri su lokalna i tabu pretraga). Stohastički metodi primenjuju slučajna pravila u procesu pretrage (primeri su simulirano kaljenje i evolutivni algoritmi). Kod determinističkih metaheuristika ista početna populacija uvek generiše isto konačno rešenje, dok kod stohastičkih metaheuristika mogu da se generišu drugačija konačna rešenja

pomoću iste početne populacije. Navedene karakteristike moraju da se uzmu u obzir kada se vrši evaluacija performansi metaheurističkih algoritama;

- metaheuristike koje su bazirane na populaciji rešenja (eng. population-based) i metaheristike koje koriste samo jedno potencijalno rešenje (eng. single-solution based). U drugu grupu spadaju metode koje u toku procesa pretrage manipulišu i vrše transformaciju samo jednog rešenja. S druge strane, populacione metaheuristike, u koje spadaju na primer optimizacija rojevima čestica i evolutivni algoritmi, vrše pretragu korišćenjem populacije potencijalnih rešenja problema. Ove dve grupe imaju komplementarne karakteristike: metaheuristike koje koriste jedno potencijalno rešenje uglavnom koriste intenzifikaciju u pretraživanju prostora, dok su populacione metaheuristike orjentisane ka diversifikaciji. Zbog komplementarnih karakteristika, ove metaheuristike se često koriste zajedno;
- metaheuristike koje koriste memoriju, tj. koje imaju mogućnost pamćenja prethodnih rešenja (eng. memory usage methods) i na one koje ne koriste memoriju, tj. koje nemaju mogućnost pamćenja prethodnih rešenja (eng. memoryless methods). Algoritmi koji ne koriste memoriju nemaju mogućnost dinamičkog estrahovanja informacija tokom procesa pretrage. Jedan od predstavnika ove vrste je malopre pomenuta metaheuristika simuliranog kaljenja. S druge strane, mnogi algoritmi koriste sistem *online* ekstrakcije memorije tokom pretrage, kao što je na primer kratkoročna i dugoročna memorija koje poseduju jedinke tabu pretrage;
- prirodom inspirisane metaheuristike (eng. nature-inspired) i na one koje nisu prirodom inspirisane (non nature-inspired). Nastanak mnogih metaheurističkih metoda je inspirisan prirodnim procesima, kao što su na primer evolutivni algoritmi i veštački imuni sistemi. Takođe su i mnogi sistemi organizama u prirodi poslužili kao inspiracija za razvoj ovih metoda, kao što na primer kolonije mrava i pčela, jata ptica i riba, itd. Osim navedenog, u literaturi se pronalaze i metaheuristike koje su inspirsane fizičkim procesima, kao što je metaheuristika simuliranog kaljenja;

Osim navedene podele, korisno je da se spomenu još dve klasifikacije. Prema jednoj

podeli, metaheuristike se dele na algoritme koji koriste samo jednu strukturu okoline i na one koje koriste skup struktura okoline. Metode optimizacije uglavnom koriste jednu strukturu okoline, ali postoje i algoritmi, kao što je metoda promenljivih okolina koja koristi skup struktura okoline. Ovakve metode se baziraju na sistematičnoj promeni okolina koja uključuje i lokalnu pretragu.

Prema drugoj podeli, metaheuristike mogu da budu konstruktivne, poboljšavajuće i hibridne. Konstruktivni algoritmi grade rešenje po principu "korak po korak" (na primer pohlepni algoritam). Poboljšavajući algoritmi nasumično biraju početno rešenje koje kasnije pokušavaju da poprave kroz niz iteracija (primer je metoda simuliranog kaljenja). Hibridni algoritmi su implementacije koje su nastale kombinacijom više algoritama.

Cela sledeća sekcija posvećena je detaljnijem prikazu metaheuristika koje nisu i koje jesu inspirisane prirodom.

2.3 Metaheuristike koje nisu inspirisane prirodom

Metaheuristike su složeniji sistemi koji se obično sastoje iz populacionih heuristika kombinovanih sa drugim tehnikama. Mogu da kreću od Monte Karlo metode, a koriste i lokalnu pretragu i gramzivi algoritam. Kao neki od predstavnika metaheuristika koje nisu inspirisane prirodom mogu da se navedu tabu pretraga i diferencijalna evolucija.

Metod Monte Karlo (eng. Monte Carlo) se u literaturi najčešće spominje kao jedna od najednostavnijih optimizacionih metoda. Ona spada u grupu stohastičkih algoritama, pošto u procesu pretrage koristi određeni nivo slučajnosti. Monte Karlo funkcioniše tako što iz skupa potencijalnih rešenja na slučajan način bira jedno. Zatim proverava da li je kvalitet slučajno izabranog rešenja bolji od do tada najboljeg pronađenog rešenja, i ako jeste, onda se to rešenje čuva i bira se novo test rešenje. Ovaj proces se ponavalja sve dok se ne zadovolji neki, unapred definisani kriterijum za prekid izvršavanja algoritma [25]. Najčešće korišćeni kriterijumi završetka su broj pokušaja popravke rešenja, tj. broj iteracija i broj pokušaja da se poboljša tekuće

najbolje rešenje. U Monte Karlo procesu verovatnoća pronalaska optimalnog rešenja raste sa povećanjem broja testiranih rešenja.

Efikasnost Monte Karlo metode najbolje može da se ilustruje na sledećem trivijalnom problemu. Neka je dat skup od n brojeva i potrebno je da se pronađe element koji pripada gornjoj polovini skupa. Prva ideja je da se izvrši komparacija $n/2$ elemenata i najveći broj među njima sigurno pripada gornjoj polovini skupa. Dakle, potrebno je da se izvrši tačno $n/2$ poređenja. Monte Karlo metod pristupa ovom problemu tako što na slučajan način bira jedan element, koga zatim proverava da li je veći od do sada pronađenog maksimuma, i ako jeste, onda se taj element proglasi za novi maksimum. Postavlja se pitanje koliko je verovatnoća da posle i koraka nije pronađen element koji pripada gornjoj polovini. Šanse da prvi izabrani element ne pripada gornjoj polovini su 0.5, a nakon drugog pokušaja ta šansa iznosi 0.5^2, itd. Jasno je da verovatnoća da nakon i koraka nije pronađen element koji pripada gornjoj polovini iznosti 0.5^i.

Na osnovu navedenog primera vidi se da se Monte Karlo metodom rešenja dobijaju sporo, a za generisanje rešenja visokog kvaliteta potrebno je da se testira veliki broj potencijalnih rešenja. Upravo iz ovog razloga Monte Karlo metod obično se ne koristi izolovano, već se kombinuje sa lokalnom pretragom, ili sa nekim drugim mehanizmom koji smanjuje slučajnost prilikom izbora sledećeg rešenja koje će da bude testirano.

Pojednostavljeni pseudo-kod Monte Karlo metode za problem minimizacije prikazan je u Algoritmu 1.

Monte Karlo metod ima veliki broj praktičnih aplikacija koje obuhvataju skoro svaku oblast. Tako na primer, ova tehnika se široko koristi u analizi neizvesnosti i osetljivosti [26].

Algoritam 1 Pseudo-kod Monte Karlo metode

inicijalizacija. Izabrati početno rešenje x i postaviti da je $x_{opt} = x$ i
da je $f_{opt} = f(x)$
while nije zadovoljen kriterijum zaustavljanja **do**
  pokušaj. Izabrati slučajno rešenje x' u prostoru dopustivih rešenja X
  provera rešenja.
  if $f(x') < f_{opt}$ **then**
    $x_{opt} = x', f_{opt} = f(x')$
  end if
end while

Lokalna pretraga (eng. local search) je metoda opšte namene koja se koristi za poboljšanje već pronađenih rešenja. Ova metoda je najefikasnija kada se traži rešenje koje ima najveću vrednost nekog kriterijuma od svih potencijalnih rešenja.

Lokalna pretraga se kreće od jednog do drugog rešenja dokle god se ne pronađe optimalno rešenje, ili dokle god nije zadovoljen neki drugi kriterijum za prekid izvršavanja algoritma. Drugim rečima, lokalna pretraga je iterativni proces, u kome se kreće od nekog početnog rešenja, a zatim se potraga preusmerava na novo rešenje u njegovoj okolini koje je potencijalno bolje. Okolina se definiše u zavisnosti od problema koji se rešava. U većini slučajeva, okolina obuhvata više od jednog rešenja, pa je potrebo da se definiše i način na koji se bira rešenje u okolini postojećeg. Najstandardniji način selekcije je metod penjanja uz brdo (eng. hill climbing). Primenom ovog metoda, pretraga se usmerava na rešenje iz komšiluka postojećeg koje ima maksimalnu vrednost određenog atributa uz očekivanje da će novo rešenje biti bolje od tekućeg.

Suštinski nedostatak lokalne pretrage je taj što se algoritam zaustavlja kada naiđe na prvi lokalni minimum, koji nije nužno i globalni optimum. Upravo zbog ovoga se ovaj i srodne metode najčešće koriste u kombinaciji sa drugim algoritmima. Tako na primer, rešenje koje pronađe gramzivi algoritam koristi se kao početni korak lokalne pretrage, ili se slučajno izabrano rešenje Monte Karlo metode proglašava za "koren" lokalne pretrage, itd.

Kada se govori o lokalnoj pretrazi, potrebno je da se spomene i metoda više-startnog

lokalnog pretraživanja (eng. multi-start local search - MLS). Ova metoda izbegava zamku zaglavljivanja u lokalnom minimumu tako što pretraga u svakoj iteraciji počinje iz novog početnog rešenja. Na ovaj način se u toku izvršavanja algoritma generiše više lokalnih minimuma, koji se zatim upoređuju i najbolji među njima se proglašava za konačno rešenje.

Gramzivi algoritam (eng. greedy algorithm) smanjuje prostor pretage, pa samim tim i složenost procesa optimizacije. Ovaj metod usmerava proces pretrage ka trenutno najboljem rešenju, ali pritom ne uzima u obzir da ovakva pretraga ne mora da vodi i ka globalnom optimumu. Dakle, gramzivi algoritam koristi relativno jednostavnu heuristiku: pronađi trenutno najbolje rešenje i idi za njim. Zbog ovoga, gramzivi algoritam ne može uvek da pronađe optimalno rešenje.

Veliki broj algoritama se bazira na principu gramzivosti, pa kada se govori o ovom algoritmu, često se misli na grupu algoritamskih koncepata. Iako gramzivi algoritam najčešće ne uspeva da pronađe optimalno rešenje, on može da na koristan način preusmeri neku drugu heuristiku ili metodu ka pravom rešenju problema [27]. Gramzivi algoritmi su jednostavni za dizajniranje i implementaciju i brzo se izvršavaju.

2.3.1 Tabu pretraga

Tabu pretraga (eng. taboo search - TS) je matematički metod optimizacije koji pripada familiji tehnika lokalne pretrage. Ovu tehniku je prvi definisao Fred W. Glover 1986. godine [28], a metoda je formalizovana tek 1989. godine.

Ukratko rečeno, TS povećava efikasnost lokalne pretrage upotrebom memorijske strukture. Koncept memorije se svodi na to da, kada se pronađe potencijalno rešenje, ono se obeležava kao "tabu" i algoritam u nastavku procesa pretrage više ne posećuje to rešenje.

Kao što je već rečeno, algoritmi lokalne pretrage se ponekada zaglavjuju u lokalnim optimumima. TS značajno poboljšava performanse lokalne pretrage putem memorijskih struktura koje opisuju već pronađena rešenja, ili pomoću seta pravila koja je zadao korisnik. Ukoliko je algoritam već ispitivao određeno potencijalno rešenje

problema u skorijem vremenskom periodu, ili ako se u formulaciji problema koristi predefinisano pravilo koje dato rešenje ne zadovoljava, onda se takvo rešenje označava kao ”tabu”. TS pamti listu od poslednjih n tabu rešenja. Tokom procesa pretrage, algoritam bira najboljeg suseda trenutnog rešenja koje se ne nalazi u listi tabu rešenja, čak iako je sused lošiji od trenutno najbolje rešenja. Na ovaj način se glavni nedostatak metode lokalne pretrage da se zaglavljuje u suboptimalnim regionima eliminiše.

Tabu liste čuvaju podatke o kretanjima rešenja za nekoliko poslednjih iteracija. Ove informacije se koriste za izbor narednog rešenja, ali i za modifikaciju definicije okoline, kako bi neka od "zabranjenih rešenja" bila izbačena iz populacije. S obzirom na činjenicu da se struktura okoline $N(x)$ rešenja x menja iz iteracije u iteraciju, metoda tabu pretrage se ubraja u grupu tehnika dinamičkog pretraživanja okoline.

2.3.2 Diferencijalna evolucija

Diferencijalna evolucija (eng. differential evolution - DE) je jedna od često korišćenih metaheurističkih metoda za rešavanje problema globalne optimizacije. Ovaj algoritam, koji se pokazao efikasnim u rešavanju brojnih zadataka, predložili su Storn i Price 1997. godine [29].

Kao i u slučaju većine drugih populacionih metaheuristika, početna slučajna populacija se generiše u fazi inicijalizacije. U svakoj generaciji izvršavanja, kreiraju se nove individue pomoću operatora ukrštanja i mutacije. Zatim se izračunava podobnost novih jedinki (eksperimentalne individue) u upoređuje se sa podobnošću starih jedinki (ciljne individue). Na osnovu rezultata ovog procesa, bolje jedinke sa zadržavaju i prenose se u sledeću generaciju. Eksperimentalne individue se generišu ukrštanjem ciljnih individua sa jedinkama koje su nastale primenom operatora mutacije (mutant individue).

Price i al. su predložili različite DE varijante [30], koje se konvencionalno nazivaju *DE/x/y/z*, gde je x niska koja označava osnovni vektor (na primer vektor koji će biti modifikovan, bilo da je on slučajan, ili najbolji u populaciji), y je broj različitih vektora pomoću kojih će se vršiti peturbacije osnovnog vektora i z označava šemu

ukrštanja koja može da bude binomna ili eksponencijalna. Klasična verzija DE algoritma ima oznaku *DE/rand/1/bi*.

Mutacija se u DE algoritmu izvodi računanjem razlike vektora slučajno odabranih individua iz iste populacije. Postoji više načina za generisanje mutant individue. Primenom najčešće korišćene strategije mutacije *DE/rand/1/bi* eksperimentalni vektor $\vec{V}_{i,g}$ se generiše dodavanjem ponderisanog vektora razlike $F(\vec{X}_{r2,g} - \vec{X}_{r3,g})$ na slučajno odabrani osnovni vektor $\vec{X}_{r1,g}$. Za svaki ciljni vektor $\vec{X}_{i,g}$, $i = 1, ..., N$, gde g označava trenutnu generaciju, a N broj individua u populaciji, mutant vektor se generiše primenom sledeće formule [31]:

$$\vec{V}_{i,g} = \vec{X}_{r1,g} + F(\vec{X}_{r2,g} - \vec{X}_{r3,g}), \tag{2.1}$$

gde su $r1, r2$ i $r3$ slučajno izabrani indeksi iz intervala $[1, N]$ koji su različiti od trenutnog indeksa i. F je faktor skaliranja iz intervala $[0, 1]$.

Eksperimentalni vektor $\vec{U}_{i,g}$ kreira se primenom operatora ukrštanja koji kombinuje komponente i-tog vektora $\vec{X}_{i,g}$ i odgovarajućeg mutant vektora $\vec{V}_{i,g}$ [31]:

$$U_{i,j,g} = \begin{cases} V_{i,j,g}, \text{ ako je } \theta \leq CR \text{ ili } j = j_{rand} \\ X_{i,jg}, \text{ u suprotnom} \end{cases} \tag{2.2}$$

gde je CR faktor ukrštanja koji uzima vrednost iz intervala $[0, 1]$ i predstavlja verovatnoću kreiranja novih vrednosti varijabli eksperimentalnog vektora na osnovu mutant vektora, dok je indeks j_{rand} slučajno izabrana diskretna vrednost iz intervala $[1, N]$. Parametar θ je slučajan broj između 0 i 1, $j = 1, ..., D$, gde je D broj parametara problema.

Konačno, da bi se odlučilo da li se novo rešenje prenosi u sledeću generaciju $g + 1$, podobnost eksperimentalnog vektora $\vec{U}_{i,g}$ se poredi sa podobnošću ciljnog vektora $\vec{X}_{i,g}$:

$$\vec{X}_{i,g+1} = \begin{cases} \vec{U}_{i,g}, \ ako \ je \ f(\vec{U}_{i,g}) < f(\vec{X}_{i,g}) \\ \vec{X}_{i,g}, \ u \ suprotnom, \end{cases} \tag{2.3}$$

Glavna prednost DE algoritma je ta što on koristi mali broj kontrolnih parametara. Kao mane navode se spora konvergencija i stagnacija populacije.

2.3.3 Ostali predstavnici metaheuristika koje nisu inspirisane prirodom

Osim navedenih algoritama, postoje i mnogi drugi algoritmi koji nisu inspirisani prirodom. Od ostalih predstavnika ove familije mogu da se navedu metoda promenljivog spusta (eng. variable neighborhood descent – VND) [32], metoda promenljivih susedstva (eng. variable neighborhood search – VNS) [32] i raspršena pretrage (eng. scatter search) [33].

Metaheuristiku promenljivih susedstva predložili su Mladenović i Hansen [32]. Ovaj algoritam izvodi proces pretrage tako što istražuje susedstva rešenja koja su dinamične i promenljive prirode. Ova metoda je efikasna u slučajevima kada su susedstva rešenja komplementarna, na primer ako lokalni optimum susedstva N_i nije lokalni optimum i susedstva N_j. Metoda promenljivog spusta je deterministička verzija metaheuristike promenljivih susedstva.

Radi boljeg razumevanja metode promenljivih susedstva prikazan je njen pseudo-kod u Algoritmu 2.

Raspršenu pretragu, kao metodu za rešavanje problema celobrojnog programiranja, predložio je Glover [33]. Ova populaciona metaheuristika početna rešenja generiše na osnovnu karakteristika prostora pretrage. Proces pretrage izvodi se sistematično i u odnosu na skup referentnih tačaka koje reprezentuju dobra rešenja u populaciji. Kvalitet rešenja ne zavisi samo od vrednosti funkcije cilja.

Algoritam 2 Pseudo-kod metode promenljivih susedstva [31]

izaberi skup struktura susedstava N_n, $n = 1, ..., n_{max}$
izaberi slučajno početno rešenje s
while kriterijum za završetak nije zadovoljen **do**
  $n \leftarrow 1$
  while $n < n_{max}$ **do**
    rukovanje: izaberi slučajno rešenje s' u n-tom susedstvu $N_n(s)$ od s
    primeni lokalnu pretragu da bi se od rešenja s' došlo do rešenja s''
    if s'' je bolje od s **then**
      $s \leftarrow s''$
      $n \leftarrow 1$
    else
      $n \leftarrow n + 1$
    end if
  end while
end while

2.4 Metaheuristike inspirisane prirodom

Metaheuristike inspirisane prirodom (eng. nature-inspired metaheuristics, bio-inspired metaheuristics) pripadaju grupi metaheurističkih optimizacionih metoda koje simuliraju prirodne procese i sisteme kada izvode aktivnosti pretrage prostora potencijalnih rešenja. U ovu grupu metaheuristika spadaju dve glavne familije, i to evolutivni algoritmi (eng. evolutionary algoritms - EA) ili algoritmi evolutivnog računarstva (eng. evolutionary computation - EC) i metaheuristike inteligencije rojeva (eng. swarm intelligence metaheuristics - SI).

Obe grupe algoritama se baziraju na sličnim principima i konceptima. Međutim, između njih postoje i značajne razlike i jedna grupa nije specijalni slučaj druge. Zbog činjenice da obe grupe algoritama imaju svoje karakteristične prednosti i nedostatke, oni se često koriste kooperativno u tzv. hibridnim pristupima, koji usvajaju prednosti jednih, uz istovremeno eliminisanje nedostataka drugih metaheuristika.

Teorija evolucije Čarlsa Darvina i zakoni nasleđivanja Gregora Mendela predstavljaju kamelj temeljac razvoja EA. Teorijska osnova za razvoj ovih algoritama pri-

kazana data je u Darvinovoj knjizi "Nastanak vrsta putem prirodnog odabiranja" [34].

Evolucija je metod pretrage velikog broja mogućnosti potencijalnih rešenja problema. Međutim, sam evolutivni proces je u suštini relativno jednostavan, pošto se vrste razvijaju i evoluiraju primenom prostih pravila poput ukrštanja, varijacije i slučajne mutacije genetskog materijala. Uz ovo se primenjuje i proces selekcije koji bira samo one jedinke koje su najbolje prilagođene okruženju u kome egzistiraju. Prilagođenost se meri na razne načine, kao što su promenljivi vremenski uslovi ili adaptacija na druge jedinke u okruženju. Kriterijumi za merenje prilagođenosti, tj. podobnosti se permanetno menjaju tokom razvoja jedinke, tako da na evoluciju može da se gleda i kao na kontinuirano traženje novih mogućnosti. Jedinke koje uspeju da opstanu propagiraju svoj genetski materijal u sledeću generaciju.

Darvin je još u svom originalnom radu isticao sledeće [34]: sve vrste su veoma plodne, i potomaka ima uvek više nego što je potrebno; veličina populacije je skoro uvek ista; količina hrane koja je na raspolaganju individuama u populaciji je ograničena; kod vrsta koje se seksualno reprodukuju postoje varijacije genetskog materijala i značajan procenat varijacija se prenose u naredne generacije mehanizmom nasleđivanja.

Na osnovu Darvinove teorije sledi da u svakoj populaciji postoji borba za preživljavanje i opstanak zbog velikog broja potomaka i ograničenih resursa poput vode, hrane i skloništa. Samo najbolje jedinke dobijaju šansu da se reprodukuju i da prenesu svoj genetski materijal u sledeće generacije.

Inspirisani procesom evolucije, EA pripadaju grupi populaciono-iterativnih metaheuristika, koje proces pretrage izvode uzastopnim izborima boljih rešenja koja se generišu pomoću operatora ukrštanja (eng. crossover) i mutacije (eng. mutation). Ukrštanjem se kombinuju rešenja (roditelji), čime se dobijaju nova, potencijalno bolja, rešenja (potomci). Mutacija je slučajna modifikacija novokreiranog rešenja. U kontekstu evolucije, podobnost (eng. fitness) jedinke se najčešće definiše kao verovatnoća da će novonastali potomak živeti dovoljno dugo da bi se reprodukovao (eng. viability) ili se određuje prema broju dece koje će imati (eng. fertility).

Pseudo-kod osnovnog EA prikazan je u Algoritmu 3.

Algoritam 3 Pseudo-kod osnovnog EA

```
početak
repeat
    izbor roditelja za ukrštanje
    ukrštanje parova roditelja i generisanje potomaka
    mutacija potomaka
    evaluacija podobnosti potomaka
    proces selekcije jedinki za učešće u sledećoj generaciji
until nije zadovoljen kriterijum zaustavljanja
kraj
```

Genetski algoritmi (eng. genetic algorithms - GA) su najstariji predstavnici metaheuristika EA.

2.4.1 Genetski algoritmi

GA su najpoznatiji predstavnici familije EA i kao takvi su najviše doprineli njihovom razvoju. Za nastanak GA zaslužan je Džon Holand, koji je teoretsku osnovu za razvoj GA postavio u svojoj knjizi "Adaptacija u prirodnim i veštačkim sistemima" (eng. Adaptation in Natural and Artificial Systems). Glavne karakteristike GA su da oni kodiraju potencijalno rešenje problema pomoću binarnih niski fiksne dužine, da se operatori ukrštanja izvršavaju nad parovima jedinki i da se operator mutacije koristi za slučajnu modifikaciju rešenja. Potencijalna rešenja problema se u kontekstu GA nazivaju hromozomi, a njihovi delovi su geni.

GA pripadaju grupi stohastičkih metoda globalnog pretraživanja koje simuliraju proces biološke evolucije. Osnovna konstrukcija je populacija individua koja se najčešće sastoji od 10 do 200 jedinki. Da bi se problem rešio pomoću GA, potencijalna rešenja (individue) je potrebno kodirati. U ovome leži jedan od najvećih izazova GA, pošto ne postoji univerzalni način kodiranja, već on zavisi od problema do problema.

Svakoj jedinki se dodeljuje određeni kvalitet, koji se računa pomoću funkcije podobnosti (eng. fitness function). GA tokom procesa pretrage stalno poboljšavaju

apsolutnu podobnost svake jedinke u populaciji i prosečnu podobnost cele populacije i na taj način proces pretrage konvergira ka optimumu. Konvergencija se postiže uzastopnom primenom genetskih operatora ukrštanja, mutacije i operatora selekcije. Procesom selekcije se favorizuju jedinke koje imaju nadprosečnu podobnost i njihovi delovi (geni), koji na taj način dobijaju veću šansu da prežive i da učestvuju kao roditelji u formiranju sledeće generacije. Individue koje su slabije prilagođene imaju manju šansu za reprodukciju i postepeno izumiru. Osnovna implikacija ovog procesa je da se nove jedinke bolje uklapaju u okolinu nego one od kojih su nastale, kao što je i slučaj u biološkim sistemima.

Najjednostavniji primer je prost GA. Njegove osnovne karakteristike su ukrštanje sa jednom prekidnom tačkom, prosta mutacija i proporcionalna, tj. rulet selekcija (eng. proportional, roulette selection).

Psuedo-kod osnovnog GA prikazan je u Algoritmu 4.

Algoritam 4 Pseudo-kod prostog GA

```
generisanje slučajne populacije od n l-bitnih hromozoma
while se ne izvrši maksimalni broj generacija do
    izračunavanje podobnosti f(x) svakog hromozoma u populaciji
    while n potomaka ne bude kreirano do
        izbor para roditeljskih hromozoma iz tekuće populacije
        sa verovatnoćom p_c izabrani roditelji se ukrštaju i kreiraju se dva potomka
        mutiranje potomaka sa verovatnoćom p_m
    end while
    zamena tekuće populacije novom populacijom
end while
```

Svaka iteracija GA naziva se generacija. GA se u najvećem broju slučajeva izvršavaju u ciklusima od 50 do 500, ili više generacija. Na kraju svakog izvršavanja (skup generacija) u populaciji se nalazi jedna ili više individua koja ima veliku podobnost. U ovom procesu veliki uticaj ima slučajnost, pa zato i rezultati različitih izvršavanja mogu značajno da se razlikuju. Zbog toga se rezultati GA prikazuju statistički, kao prosečne vrednosti više izvršavanja algoritma nad istim problemom i sa istim podešavanjima parametara. Vrednosti parametara veličine populacije n, verovat-

noće ukrštanja i mutacije, p_c i p_m, respektivno utiču u velikoj meri na efikasnost i efektivnost GA, pa time i na dobijene rezultate.

2.4.2 Simulirano kaljenje

Simulirano kaljenje (eng. simulated annealing - SA) je metaheuristika inspirisana prirodom za rešavanje problema globalne optimizacije. SA simulira fizički proces kaljenja metala, tj. proces zagrevanja i kontrolisanog hlađenja metala u stanje kristalne rešetke sa najmanjom energijom i većom veličinom kristala. Čvrstina kristalne rešetke odgovara funkciji cilja koja se optimizuje, a način hlađenja emulira proces pretrage. Prilikom procesa kaljenja, potrebno je pažljivo odrediti temperaturu i stopu zamrzavanja (eng. cooling rate), koja se u literaturi često naziva raspored kaljenja.

Prve aplikacije metaheuristike SA su prikazali Kirkpatrik, Gelat i Veći 1983. godine [35]. Od tada u literaturi može da se nađe veliki broj primera primena ove metode. Za razliku od nekih drugih, posebno determinističkih metoda optimizacije, SA ima veliku prednost zato što izbegava zamku zaglavljivanja u lokalnom minimumu. Tačnije, dokazano je da SA konvergira ka globalnom optimumu, ako se u procesu pretrage koristi dovoljno slučajnosti u kombinaciji sa sporim hlađenjem.

SA može da se uporedi sa bacanjem loptice. Kada se loptica baci, ona će nekoliko puta da odskoči sve dok ne izgubi energiju i na kraju će da se ”smiri” u nekom od lokalnih minimuma. Ako loptica ima dovoljno energije da može više puta da odskače, uz postepeni gubitak energije, na kraju će da se "smiri" u tački koja predstavlja globalni optimum.

Glavna ideja metaheuristike SA je da se koristi slučajna pretraga i da osim prihvatanja promenjenih rešenja koja poboljšavaju funkciju cilja, prihvataju se i rešenja koja nisu idealna. Tako na primer, u slučaju problema minimizacije, svaka promena rešenja koja smanjuje vrednost funkcije cilja f će biti prihvaćena, ali će isto tako biti prihvaćene sa određenom verovatnoćom p i neke promene koje povećavaju vrednost funkcije f. Ova verovatnoća, koja se u literaturi naziva tranziciona verovatnoća se određuje kao:

$$p(i) = e^{-\frac{\delta E}{kT}}, \tag{2.4}$$

gde je k Bolcmanova konstanta, a T temperatura koja kontroliše proces kaljenja. δE predstavlja promenu u energetskom nivou. Ovako definisana verovatnoća tranzicije se bazira na Bolcmanovoj distribuciji iz fizike. Najprostiji način da se poveže δE sa promenom vrednosti funkcije cilja δf je prikazan sledećom jednačinom:

$$\delta E = \gamma \delta f, \tag{2.5}$$

gde je γ realna konstanta. Radi jednostavnosti, može da se postavi da je $k = 1$ i $\gamma = 1$. U ovom slučaju, jednačina kojom se računa verovatnoća p postaje:

$$p(\delta f, T) = e^{-\frac{\delta f}{T}} \tag{2.6}$$

Bez obzira da li se promena rešenja prihvata, ili se ne prihvata, obično se koristi pseudo-slučajan broj r kao granična vrednost prihvatanja promene, tako da, ukoliko je $p > r$ ili [36]:

$$p = e^{-\frac{\delta f}{T}} > r, \tag{2.7}$$

rešenje će biti prihvaćeno.

Metaheuristika SA može da se prikaže sledećim jednostavnim pseudo-kodom [36].

Algoritam 5 Pseudo-kod metaheuristike SA

početak

funkcija cilja $f(x)$, $x = (x_1, ..., x_p)^T$

inicijalizuj početnu temperaturu T_0 postavi da je najbolje rešenje x^0

postavi konačnu temperaturu T_f i maksimalni broj iteracija N

definiši raspored hlađenja $T \rightarrow \alpha T, (0 < \alpha < 1)$

while $T > T_f$ i $n < N$ **do**

pomeri rešenje na slučajnu novu lokaciju $x_{n+1} = x_n + rnd$

izračunaj $\delta f = f_{n+1}(x_{n+1}) - f_n(x_n)$

prihvati novo rešenje ako je bolje od starog

ako novo rešenje nije bolje, onda generiši slučajan broj r

prihvati novo rešenje ako je zadovoljen uslov iz jednačine (2.7)

ažuriraj pozicije najboljeg rešenja x_* i f_*

end while

kraj

2.4.3 Inteligencija rojeva

Metaheuristike inteligencije rojeva su noviji predstavnici algoritama inspirisanih prirodom, zbog činjenice da mogu da generišu brza, jeftina i kvalitetna rešenja teških optimizacionih problema. Oblast algoritama rojeva je živa i dinamična oblast, čemu ide u prilog veliki broj objavljenih radova u vrhunskim svetskim časopisima.

Metaheuristike inteligencije rojeva su centralna tema ovog rada i zato je njima posvećeno celo sledeće poglavlje.

3 Metaheuristike inteligencije rojeva

3.1 Pojam i definisanje metaheuristika inteligencije rojeva

Način na koji se grupe organizama ponašaju u prirodi glavni je izvor inspiracije za nastanak metaheuristika inteligencije rojeva. Proučavanjem rojeva pčela, kolonija mrava, jata ptica i riba, itd., uočeno je da grupa individua pokazuje neka ponašanja koja pojedinačni članovi ne ispoljavaju. Primećeno je da su roj i jato kao celina inteligentniji od svih članova koji im pripadaju. Upravo ova činjenica predstavlja osnovu za nastanak algoritama inteligencije rojeva.

Roj je grupa koja se sastoji od velikog broja homegenih i jednostavnih agenata koji interaguju međusobno i sa svojim okruženjem, bez prisustva centralne komponente koja koordinira njihovo globalno ponašanje. Algoritmi inteligencije rojeva, kao deo familije prirodom inspirisanih metaheuristika, su kroz naučnu literaturu nedavno postali poznati kao populacioni algoritmi koji mogu da proizvedu brza i kvalitetna rešenja velikog broja složenih problema [37].

Postoji više definicija algoritama rojeva. U opštem smislu, algoritmi inteligencije rojeva mogu da se definišu kao relativno nova grana veštačke inteligencije (eng. artificial intelligence - AI), koja se modeliranjem grupnog ponašanja rojeva u prirodi, poput mravljih kolonija, rojeva pčela, jata ptica i riba, itd., koristi za rešavanje problema. Bez obzira na činjenicu da su agenti (insekti ili članovi jata) relativno nesofisticirani i ograničenih sposobnosti, oni putem međusobnih interakcija sa jedinstvenim obrascima u ponašanju kooperativno rešavaju zadatke koji su neophodni za njihov opstanak.

Metaheuristike rojeva temelje se na sledećim konceptima [38]: ponašanje grupe koje je rezultat kooperativnog ponašanja individua; individualni agenti su uglavnom homogeni, a njihove interakcije su asinhrone i paralelne; centralizovana kontrola ne postoji, ili postoji u maloj meri; agenti komuniciraju i putem indirektnih interakcija u formi stigmetrije; "korisno ponašanje" je relativno jednostavno, kao što je na primer pronalaženje dobrog mesta za hranu, preživljavanje u dinamičkom okruženju i

pronalaženje dobrog gnezda.

Interakcije između individua u roju mogu da budu direktne ili indirektne. Direktne interakcije najčešće se odnose na situacije kada je individua u audio ili vizuelnom dometu drugih individua, kao što je na primer privlačenje svitaca na osnovu emitovanog intenziteta svetlosti. S druge strane, individue su u indirektnom kontaktu u situacijama kada jedna individua promeni okruženje, a druga reaguje na te promene. Takav je slučaj na primer sa algoritmom koji simulira ponašanje mrava, gde se indirektna komunikacija između mrava odvija preko feromona, koga mravi ostavljaju tokom kretanja. Ovakav vid interakcije naziva se stigmetrija i označava komunikaciju između individua preko okruženja u kome egzistiraju [39].

Jedan od važnijih aspekata ponašanja rojeva koji se modelira računarskim algoritmima je sposobnost samoorganizacije. Samoorganizacija je sposobnost grupe individua da izvrše zadatak i reše problem bez prisustva centralne kontrole. Drugim rečima, jedinke svoje ponašanje usmeravaju na osnovu lokalnih informacija, bez poznavanja globalnog obrasca. Četiri osnovne komponente samoorganizacije su [38]:

1. *pozitivna povratna sprega (eng. positive feedback).* Dobar primer je izbor rešenja za eksploataciju od strane pčela posmatrača na osnovu informacija o kvalitetu rešenja dobijenih od pčela radilica;
2. *negativna povratna sprega (eng. negative feedback).* Ova sprega služi kao kontrateg pozitivnoj povratnoj sprezi radi uspostavljanja balansa i pomaže u stabilizaciji kolektivnog obrasca u formi zasićenja ili isrpljenja resursa. Primer su pčele radilice koje napuštaju izvor hrane koji više nema nektara;
3. *slučajnost (eng. randomness).* Ova osobina omogućava pronalaženja novih rešenja. Kao primer može da se navede izgubljeni mrav koji slučajno pronalazi novi izvor hrane, ili pčela izviđač koja bira slučajni cvet za eksploataciju i
4. *višestruke interakcije (eng. multiple interactions).* Samo je jedna jedinka dovoljna da se napravi samoorganizirajuća struktura, kao što je stabilan trag od feromona u slučaju mravljih kolonija, ali je najčešće potreban minimalan broj jedinki koje međusobno interaguje da bi algoritam dao zadovoljavajuće rezultate.

Da bi metaheuristike rojeva što bolje modelirale prirodne sisteme, usvojeni su sledeći principi [37]: princip bliskosti, princip kvaliteta, princip različitosti odgovora i principi stabilnosti i prilagođavanja.

Princip bliskosti (eng. proximity principle). Članovi roja su sposobni za jednostavna izračunavanja koja zavise od okruženja u kome se nalaze. U ovom slučaju, izračunavanja se smatraju za neposredne odgovore, kroz ponašanje agenata, na promene u okruženju, kao što su promene koje su izazvane iterakcijama između agenata. Odgovori se razlikuju i zavise od kompleksnosti agenata. Međutim, osnovna ponašanja su zajednička za sve agente, kao što su na primer potraga za resursima neophodnim za preživljavanje i izgradnja gnezda;

Princip kvaliteta (eng. quality principle). Osim osnovnih sposobnosti izračunavanja, roj je u stanju da odgovori na kvalitativne faktore, kao što su hrana i bezbednost;

Princip različitosti odgovora (eng. diverse response principle). Resursi najčešće nisu koncentrisani u uskom regionu. Ova distribucija bi trebala da bude dizajnirana tako da je svaki agent maksimalno zaštićen od fluktuacija u okruženju.

Principi stabilosti i prilagođavanja (eng. stability and adaptability principles). Od roja se očekuje da se prilagođava promenama u okruženju bez nagle promene u načinima svog ponašanja, zato što svaka promena snosi troškove u vidu potrošene energije.

3.2 Osnovni teorijski principi metaheuristika rojeva

Metaheuristike inteligencije rojeva su iterativni algoritmi. Jednačina iteracije (1.31) važi za determinističke algoritme. Međutim, u savremenim metaheurističkim metodama najčešće se koristi slučajnost, tako što se u algoritam uvodi m slučajnih varijabli $\epsilon = (\epsilon_1, ..., \epsilon_m)$. Tako na primer u SA pristupu koristi se jedna slučajna varijabla, dok algoritam optimizacije rojevima čestica koristi njih tri.

Osim slučajnih varijabli, metaheuristike rojeva koriste i skup od k parametara koji se predstavlja u formi vektora parametara $p = (p_1, ..., p_k)$. U opštem slučaju, metaheuristika roja sa k parametara i m slučajnih varijabli može da se definiše kao:

$$x^{t+1} = A(x^t, p(t), \epsilon(t)), \tag{3.1}$$

gde je A funkcija nelinearnog mapiranja trenutnog rešenja (d-dimenzioni vektor x^t) na novo rešenje x^{t+1}.

Algoritmi inteligencije rojeva temelje se na principima samoorganizacije. Postoje uslovi koji moraju biti zadovoljeni da bi samoorganizacija mogla da se održi u složenom sistemu, kakav je sistem metaheuristika rojeva [18]: sistem mora da bude dovoljno veliki i da podržava veliki broj različitih stanja; u sistemu mora da postoji dosta različitosti, kao što su petrubacije i šumovi, ili da se nalazi daleko od ravnotežnog stanja; sistem može da se razvija dug vremenski period i u sistemu mora da deluje mehanizam selekcije, ili neki drugi zakon.

Drugim rečima, sistem sa stanjima S će se razviti ka samoorganizovanom stanju S_*, vođen mehanizmom $\alpha(t)$ sa skupom parametara α [18]:

$$S \xrightarrow{\alpha(t)} S_* \tag{3.2}$$

Metaheuristika roja definisana jednačinom (3.1) je samoorganizirajući sistem koji počinje proces pretrage sa mnogo mogućih stanja x^t i pokušava da konvergira ka optimalnom rešenju/stanju x_*, a pritom je vođen algoritmom A [18]:

$$f(x^t) \xrightarrow{A} f_{min}(x_*) \tag{3.3}$$

Metaheuristike inteligencije rojeva takođe mogu biti analizirane sa aspekta načina na koji pretražuju prostor pretrage. Svi algoritmi inteligencije rojeva moraju da imaju dve osnovne komponente - eksploataciju i eksploraciju - koje se takođe nazivaju i intenzifikacija i diversifikacija. U nastavku rada se ova dva para termina koriste kao sinonimi.

Proces eksploatacije (intenzifikacije) koristi sve raspoložive informacije o problemu, kako bi pomogao u generisanju novih rešenja koja su bolja od trenutnih. Međutim, ovaj proces, kao i informacije na osnovu kojih se izvodi, su lokalnog karaktera.

Intenzifikacija obično vodi ka visokim stopama konvergencije. Međutim, proces pretrage može da se zaglavi u suboptimalnim regionima zato što pozicija konačnog rešenja u velikoj meri zavisi od početne pozicije.

S druge strane, proces eksploracije (diversifikacije) vrši efikasniji proces pretrage i generiše različita rešenja koja su daleko od trenutnih rešenja. Dakle, ovaj proces sprovodi globalnu pretragu. U slučaju diversifikacije postoji samo minimalna verovatnoća da će algoritam da se zaglavi u prostoru lokalnih optimuma i globalni optimum je pristupačniji. Eksploracija ima slabu konvergenciju i troši mnogo izračunavanja na rešenja koja se nalaze daleko od optimuma.

Da bi algoritam inteligencije rojeva dao zadovoljavajuće rezultate, odgovarajući balans između eksploatacije i eksploracije mora da se uspostavi. Algoritam koji koristi mnogo intenzifikacije a malo diversifikacije mogao bi brzo da konvergira, ali uz malu verovatnoću da će da pronađe globalni optimum. S druge strane, ako se koristi velika snaga eksploracije sa malom eksploatacijom, algoritam sa većom verovatnoćom pronalazi optimalni domen, ali neće konvergirati brzo. Kao dva ekstremna primera mogu da se navedu Monte Karlo i metoda nizbrdice (eng. downhill method). Čista eksploracija teži ka Monte Karlu, dok čista eksploatacija teži prema determinističkoj metodi nizbrdice.

Metaheuristike rojeva u većoj ili manjoj meri koriste i operatore evolutivnih algoritama, i to selekciju, ukrštanje i mutaciju. Operator ukrštanja se uglavnom koristi za pretragu u okviru jednog poddomena i omogućava bržu konvergenciju. Mutacija je glavni mehanizam za izbegavanje lokalnih domena i koristi se za uvođenje slučajnosti u proces pretrage. Operator selekcije je pokretačka snaga procesa pretrage koja omogućava sistemu da se razvije u željena stanja. U većini implementacija selekcija izvodi jaku intenzifikaciju.

Pretragom literature zapaženo je da veliki broj implementacija algoritama rojeva koristi trenutno najbolje rešenje u izvođenju procesa pretrage. Međutim, uloga koju ovo rešenje ima može biti "mač sa dve oštrice". S jedne strane, trenutno globalno najbolje rešenje g^* može da ubrza konvergenciju algoritma zato što poseduje važne informacije o optimizacionom problemu. Međutim, na osnovu empirijskih i

matematičkih analiza zaključuje se da prekomerna upotreba ovog rešenja u procesu pretrage može da implicira preuranjenu konvergenciju zato što trenutno najbolje rešenje nije nužno i pravi globalni optimum.

3.3 Glavni predstavnici metaheuristika rojeva

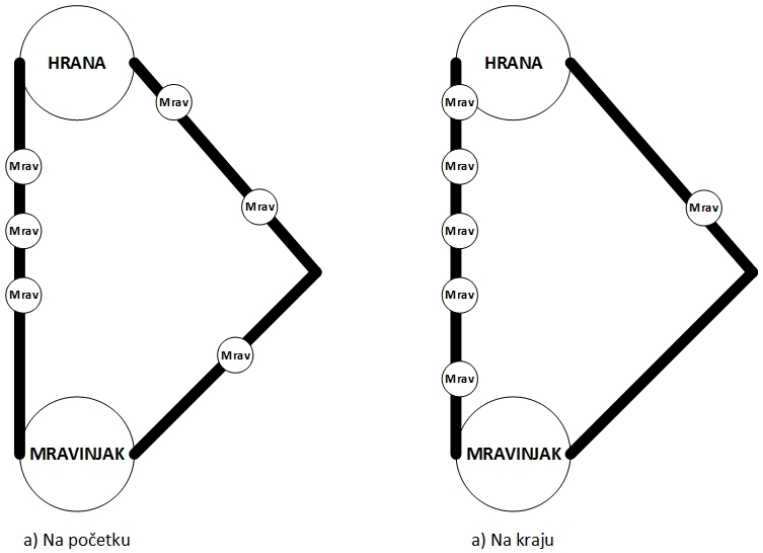
Posmatrano kroz vremensku prizmu, koreni koncepta metaheuristika rojeva nalaze se u ponašanju jata ptica i kolonija mrava. Prva dva algortima iz ove grupe, mravlji algoritmi i algoritam optimizacije rojevima čestica, emulirali su ove grupe organizama. Tvorci mravljeg algoritma su Dorigo i Colorni [40], dok su Kennedy i Eberhart u svom radu uveli algoritam optimizacije rojevima čestica [41]. Kasnije su nastali i mnogi drugi algoritmi koji simuliraju ponašanje grupe organizama iz prirode, kao što su rojevi pčela, jata riba, grupa kukavica, svitaca i račića, kolonije bakterija, itd.

3.3.1 Optimizacija mravljim kolonijama

Algoritam optimizacije mravljim kolonijama (eng. ant colony optimization - ACO) najčešće se koristi za rešavanje grafovskih problema. ACO je prvo predložen za TSP [39].

Svaki mrav tokom svog kretanja ostavlja supstancu koja se naziva feromon. Mravi indirektno komuniciraju preko feromona tako što se kreću u pravcu jačeg feromonskog traga. Jedan primer ovakvog ponašanja je nalaženje kraćih putanja od mravinjaka do izvora hrane. U početku se mravi kreću dosta haotičnim putanjama. Kako vreme prolazi, sve veći broj mrava se kreće kraćom putanjom od mravinjaka do izvora hrane, a u na kraju skoro svi. Primer je dat na Slici 3.1.

Ovakvo ponašanje je posledica toga što izbor putanje zavisi od količine feromona na određenoj stazi. Feromon isparava, tako da na dužim putanjama, za koje je mravima potrebno više vremena da prođu, ispari više feromona nego na kraćim. Zbog toga, kada mrav sledeći put bira putanju, veće su šanse da izabere kraći put. Na kraju skoro svi mravi koriste kraće putanje zato što se na njima nalazi veća količina feromona.



Slika 3.1: Primer pronalaska kraćeg puta od mravinjaka do izvora hrane

Ponašanje mrava pretvara se u matematički model koji se se sastoji iz sledećih komponenti: pravilo izbora koje pojedinačni mravi koriste za izbor sledećeg dela puta, globalno pravilo ažuriranja feromonskog traga i lokalno pravilo ažuriranja koje simulira isparivanje feromona.

Globalno pravilo ažuriranja može da se prikaže jednačinom [39]:

$$\tau_{ij} = (1 - \gamma)\tau_{ij} + \gamma\Delta\tau^k, \ \forall ij \in B^k, \tag{3.4}$$

gde je τ_{ij} feromonski trag koji se menja, γ je fiksni parametar koji određuje uticaj novih rešenja, a $\Delta\tau^k$ predstavlja kvalitet rešenja koju je postigao k-ti mrav. B^k je skup koji se sastoji od svih ivica u rešenju koje je konstruisao k-ti mrav.

Proces lokalnog ažuriranja definiše se kao [39]:

$$\tau_{ij} = (1 - \delta)\tau_{ij} + \delta\tau_0, \tag{3.5}$$

gde je δ predefinisana konstanta iz intervala $[0, 1]$, a τ_0 predstavlja kvalitet rešenja koje je generisano pomoću gramzivog algoritma u kom je heuristička funkcija data dužinom ivice na grafu.

Pseudo-kod osnovnog ACO algoritma dat je u Algoritmu 6.

Algoritam 6 Pseudo-kod osnovnog ACO algoritma

```
while dok nije kraj do
    for svakog mrava do
        kreiraj potencijalno rešenje
        vrednuj rešenje
    end for
    odaberi podskup mrava
    for sve mrave iz odabranog skupa do
        ažuriraj tragove feromona
    end for
    isparivanje feromonskih tragova
end while
```

3.3.2 Optimizacija rojevima čestica

Optimizacija rojevima čestica (eng. particle swarm optimization - PSO), iniciran je pokušajima da se na računaru modelira kretanje jata ptica. Dva istraživača, Eberhart i Kennedy su uspela da pravila kretanja jata ptica uspešno inkorporiraju u optimizacioni algoritam. Svoje ideje i koncepte prikazuju u dva sveobuhvatna rada iz 1995. godine [41], [42]. Mehanizam pretrage ovog algoritma se izvodi na osnovu najboljeg pronađenog rešenja i njegovog suseda.

Svaka čestica u populaciji je definisana svojom pozicijom i brzinom (eng. velocity). PSO algortiam ima dualno memoriju - pamti najbolje rešenje koje su napravile sve čestice i najbolje rešenje koje je generisano od strane svake čestice pojedinačno. Promena položaja jedinki u populaciji modelira proces pretrage. Jedinke menjaju svoj položaj na osnovu sopstvenog iskustva i na osnovu iskustava svojih bliskih suseda, čime se emuliraju socijalne interakcije između jedinki. Smer kretanja svake jedinke se određuje na osnovu najboljeg pronađenog rešenja individue (individualni faktor) i na osnovu najboljeg pronađenog rešenja jedinki koje su u njenoj neposrednoj okolini (socijalni faktor). Proces pretrage se određuje uticajem svakog od ova dva faktora. Ako veći uticaj ima individualni faktor, onda je proces istraživanja (eksploracije) prostora pretrage veći. S druge srane, ako socijalni faktor dominira, onda se mehanizam pretrage uglavnom usmerava procesom intenzifikacije (eksploatacije). PSO

na ovaj način vrši balansiranje globalnog i lokalnog pretraživanja, čime se vrši fino pretaživanje prostora potencijalnih rešenja problema.

PSO algoritam se izvršava u iteracijima kroz tri koraka, sve dok se ne ispuni uslov završetka algoritma (eng. termination condition): izračunavanje vrednosti funkcije podobnosti svake čestice; ažuriranje individualne i globalne vrednosti funkcije podobnosti; ažuriranje brzine i položaja svake jedinke u populaciji.

Metaheuristika PSO ukratko može da se opiše na sledeći način:

1. početna populacija se inicijalizuje normalnom raspodelom na celom skupu rešenja;
2. svakoj čestici u populaciji se dodeljuje početna brzina i pravac kretanja;
3. određuje se najbolja čestica;
4. svakoj čestici se koriguje pravac kretanja na osnovu položaja jedinki u njenoj okolini i neke slučajne vrednosti. Dodatno se popravlja pravac kretanja u zavisnosti od pozicije najboljeg rešenja (čestice), tj. kretanje čestice se usmerava u pravcu najboljeg rešenja;
5. koraci 2,3 i 4 se ponavljaju sve dok se ne postigne rešenje zadovoljavajućeg kvaliteta, odnosno izvrši maksimalni broj dozvoljenih koraka.

3.3.3 Optimizacija pretragom kukavica

Optimizacija pretragom kukavica (eng. cuckoo search - CS) spada u grupu novijih algoritama rojeva, kojeg su prikazali Yang i Deb 2009. godine [43]. Osnovni princip algoritma bazira se na parazitima gnezda nekih vrsta kukavica, gde se potencijalna rešenja problema prikazuju kao gnezdo. Skorašnje studije pokazuje da je CS mnogo efikasnija metoda od PSO i GA [44].

Kukavice imaju mnoge karakteristike koje ih razlikuju od drugih vrsta ptica. Pre svega, kukavice koriste agresivnu strategiju razmnožavanja. Neke vrste kukavica čak ležu svoja jaja u gnezda drugih ptica, i uklanjaju jaja ptica domaćina, kako bi povećali verovatnoću izleganja svojih jaja. Ova pojava naziva se paraziti gnezda.

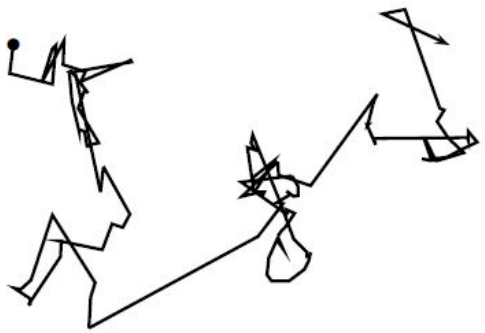
Posmatranjem putanja kojom se životinje kreću tokom lova, može da se zaključi

da se potraga za hranom vrši na slučajan, ili pseudo-slučajan način. Trajektorija je slučajan hod (eng. random walk) zato što se sledeći korak određuje na osnovu trenutne lokacije i verovatnoće da će da se pređe na sledeću lokaciju. Zbog toga je proces pretrage metaheuristike CS dodatno poboljšan takozvanim Lévy letovima, kao jednom od metoda slučajnog hoda, gde se dužina koraka određuje na osnovu deformisane raspodele (eng. heavy-tailed distribution). Na Slici 3.2 prikazan je primer putanje Lévy leta od 50 koraka. Više o Lévy letovima može se naći u [44].

Da bi se ponašanje kukavica što lakše modeliralo pomoću računarskog algoritma, primenjuju se pravila koja aproksimiraju realan sistem [45]. Generisanje novog rešenja i-te kukavice primenom Lévy leta (jendačina pretrage) može da se prikaže kao [45]:

$$x_i^{t+1} = x_i^t + \alpha \oplus Levy(\lambda), \tag{3.6}$$

gde je $\alpha > 0$ veličina koraka, a t i $t+1$ predstavljaju trenutnu i sledeću iteraciju, respektivno. Simbol $\oplus$ označava skalarni proizvod.



Slika 3.2: Putanja Lévy leta od 50 koraka [44]

Dužina slučajnog koraka izvodi se na osnovu Lévy raspodele sa beskonačnom disperzijom i srednjom vrednošću [44]:

$$Levy\ u = t^{-\lambda},\ (1 < \lambda \leq 3), \tag{3.7}$$

U ovom slučaju uzastopni koraci (skokovi) kukavica formiraju slučajan hod koji prati dužinu koraka koja se izvodi primenom zakona kvadrata sa deformisanom ras-

podelom. Neka od novih rešenja mogu da se generišu u okolini najbolje pronađenih rešenja, čime se ubrzava lokalna pretraga. Međutim, većina rešenja bi trebalo da budu generisana daleko od trenutno najboljeg rešenja, da bi se sprečilo da se algoritam zaglavi u nekom od suboptimalnih regiona.

Pseudo-kod metaheuristike CS dat je u Algoritmu 7. Uslov za prekid izvršavanja algoritma najčešće se postavlja kao maksimalan broj generacija, mada mogu da se koriste i drugi uslovi.

Algoritam 7 Pseudo-kod CS algoritma

početak
definiši funkciju cilja $f(x)$, $x = (x_1, x_2, ..., x_d)^T$
$t = 0$
generiši početnu populaciju od n gnezda ptica domaćina $x_i, i = 1, 2, ..., n$
while $t < maxGen$ **do**
  for all sve kukavice u populaciji **do**
    pomeri kukavicu na slučajnu lokaciju pomoću Lévy leta
    izračunaj vrednost funkcije podobnosti svake kukavice F_i
    izaberi slučajno gnezdo iz skupa raspoloživih gnezda, npr. j
    if $F_i > F_j$ **then**
      zameni rešenje j novim rešenjem
    end if
  end for
  procenat p_d starih gnezda se napušta i grade se nova gnezda
  sačuvaj najbolje rešenje ili gnezda sa najboljim rešenjima
  sortiraj sva rešenja prema podobnosti i nađi trenutno najbolje rešenje
end while
izvrši dodatna izračunavanja i vizualizuj rešenja
kraj

3.3.4 Optimizacija jatom planktona

Optimizacija jatom planktona (eng. krill herd - KH) spada u grupu novijih metaheurističkih algoritama koju su prvi predložili Gandomi i Alavi [46]. Ponašanje zooplanktona u prirodi poslužilo je kao glavna inspiracija za razvoj ovog algoritma. Plankton označava zajednicu živih organizama koji plutaju jer nemaju fizičke mo-

gućnosti za kretanje. Zbog fizičkih i hemijskih procesa, pomoću kojih ove grupe organizama međusobobno komuniciraju, jata planktona pokazuju karakteristike kolektivnog kretanja i formiranja grupa. Ova pojava naziva se društveno ponašanje.

Kada grabljivice napadnu jato planktona, one prvo uklanjaju individualne organizme, što za posledicu ima smanjivanje gustine jata. Stvaranje jata od pojedinačnih planktona je proces sa više ciljeva, a dva glavna su: povećanje gustine jata i pronalaženje hrane. Pozicija svake individue u jatu zavisi od tri faktora [46]: kretanja drugih individua jata, aktivnosti ishrane i slučajne difuzije.

Budući da optimizacioni algoritam mora da bude u stanju da pretražuje prostor bez obzira na njegovu dimenziju, u implementaciji KH algoritma Langranžov model je generalizovan u n-dimenzionom prostoru pretrage [46]:

$$\frac{dX_i}{d_t} = N_i + F_i + D_i, \tag{3.8}$$

gde N_i predstavlja kretanje pod uticajem drugih individua u jatu, F_i označava kretanje radi traženja hrane, dok D_i označava fizičku difuziju i-og planktona u populaciji.

Na primer, kretanje planktona pod uticajem drugih članova populacije prikazuje se sledećom jednačinom [46]:

$$N_i^{new} = N^{max}\alpha_i + \omega_n N_i^{old}, \tag{3.9}$$

gde je

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target}, \tag{3.10}$$

U prikazanim jednačinama, N^{max} je najveća brzina kretanja, w_n je inerciona težina kretanja čija je vrednost u ospegu $[0, 1]$, N_i^{old} je brzina kretanja prethodne pozicije, α_i^{local} je lokalni efekat izazvan od strane individua iz susedstva i α_i^{target} označava ciljni smer kretanja koji se određuje prema najboljoj individui u populaciji.

Pseudo-kod metaheuristike KH prikazan je u Algoritmu 8.

Algoritam 8 Pseudo-kod KH algoritma

I Definicija: definiši optimizacioni problem, granice parametara, itd.

II Inicijalizacija: kreiranje početne populacije rešenja

III Izračunavanje podobnosti: evauliraj sve planktone na osnovu njihove trenutne pozicije

IV Kretanje svake individue na osnovu:

a) pozicije drugih individua

b) kretanja radi ishrane

c) fizičke difuzije

V Implementacija genetskih operatora

VI Ažuriranje: ažuriranje pozicije svih individua u populaciji

VII Ponavljanje: Idi na korak III dok se ne ispuni kriterijum za završetak izvršavanja algoritma

IX Kraj

3.3.5 Imunološki algoritmi

Algoritmi iz porodice imunoloških algoritama (eng. immunological algorithms – IA) bave se razvojem modela, apstrahovanjem imunog sistema i njegovom primenom u algoritmima. Ova grupa metaheuristika radi sa populacijom antitela, gde svako antitelo predstavlja potencijalno rešenje problema koji se optimizuje, dok antigen reprezentuje funkciju cilja čiji se optimum traži.

Jednostavni imunološki algoritam (eng. simple immunological algorithm – SIA) razvili su 2002. godine Cutello i Nicosia [47]. SIA koristi binarno kodiranje rešenja, gde se svako rešenje predstavlja l-bitnim nizovima. Afinitet antitela prema antigenu predstavlja se kvalitetom, tj. funkcijom podobnosti.

Algoritam selekcije klonova (eng. clonal selection algorithm - CLONALG) je još jedan predstavnik ove grupe metaheuristika [48]. Osnovna ideja algoritma CLONALG slična je kao i kod SIA. Svako antitelo klonira se u obimu koji je proporcionalan njegovom afinitetu. Klonovi se izlažu procesu hipermutacije, čime se kreira nova populacija. Intenzitet hipermutacije je obrnuto proporcionalan afinitetu antitela. Glavna razlika u odnosu na SIA je ta što se antitela kodiraju kao nizovi od l realnih brojeva, i zato svako antitelo može da se posmatra kao tačka u l-dimenzionom prostoru.

Metaheuristika veštačkog imunološkog sistema (eng. artificial immune system – AIS) je populaciona, prirodom-inspirisana metoda koja se uglavnom koristi za numeričku optimizaciju [49]. Ovaj algoritam, koga su izmislili Coello Carlos i Cruz Cortes, jedan je od prvih imunoloških algoritama koji je korišćen za rešavanje problema optimizacije funkcija. Funkcija koja se optimizuje modelira se kao antigen, a populacija potencijalnih rešenja predstavlja populaciju antitela. AIS algoritam i njegove modifikovane verzije uspešno su primenjivane na probleme neprekidne i diskretne optimizacije [50].

3.3.6 Kratak prikaz algoritama rojeva koji su hibridizovani

Ovaj rad se detaljnije bavi algoritmima rojeva koji su hibridizavni. Zbog toga su metaheuristike inteligencije rojeva koje su korišćene u hibridizacijama ovde samo ukratko prikazane. Detaljni opisi dati su u Poglavlju 5.

Jedna od najpoznatijih grupa algoritama inteligencije rojeva je familija algoritama koji emuliraju ponašanje rojeva pčela, a jedan od najšire korišćenih među njima je algoritam veštačke kolonije pčela (eng. artificial bee colony - ABC). Ovaj pristup je prvi predložio Karaboga [51] za probleme neprekidne optimizacije bez ograničenja, a kasnije su ga razvili Karaboga i Bastuk [52], [53]. Pristup se bazira na tri grupe pčela koje izvode procese intenzifikacije i diversifikacije prostora pretrage.

Algoritam optimizacije pretragom svitaca (eng. firefly algorithm - FA) pripada grupi novijih pristupa metaheuristika rojeva, a inspirisan je svetlosnom signalizacijom svitaca. Tvorac ovog algoritma je Xin-She Yang, koji je u svom radu prikazao optimizaciju neprekidnih problema bez ograničenja pomoću ove metaheuristike [54]. Glavna ideja ovog algoritma je da se svaki svitac u populaciji kreće ka poziciji svetlijeg svica, dok intenzitet svetlosti svakog svica zavisi od njegove podobnosti. Ukoliko ne postoji svetliji svitac u populaciji, svitac se kreće pseudo-slučajnom putanjom.

Metaheuristika optimizacija pretragom tragača (eng. seeker optimization algorithm - SOA) emulira ponašanje grupe ljudi koji timski (kooperativno) izvode proces pretrage. Izumitelji ovog algoritma su Dai, Chen i Zhu. Algoritam je u početku testiran na problemima globalne optimizacije bez ograničenja složenih funkcija [55],

[56]. SOA je skoro promenila ime od strane svog pronalazača Chaohua Dai na metaheuristiku optimizacije grupe ljudi (eng. human group optimizer - HGO) [57].

Algoritam slepog miša (eng. bat algorithm - BA) razvio je Xin-She Yang 2010 godine [58]. BA modelira sposobnost eholokacije slepih miševa koji detektuju i izbegavaju prepreke pomoću zvučnih odjeka. Kao i svi drugi prirodom inspirisani algoritmi, BA aproksimira realan sistem koga modelira.

Algoritam vatrometa (eng. fireworks algorithm - FWA) inspirisan je procesom eksplozije vatrometnih raketa. Kada raketa eksplodira, čitav spektar varnica prekrije nebo oko tačke gde se desila eksplozija. Proces eksplozije može da se posmatra i u kontekstu pretrage lokalnog prostora pomoću varnica, koje se generišu eksplozijom vatrometne rakete. [59]. FWA je relativno nova metaheuristika koju su kreirali Tan i Zhu 2010. godine [59].

3.4 Pregled metaheuristika rojeva za probleme globalne optimizacije

Pregledom literature vidi se da postoji veliki broj implementacija metaheuristika rojeva, kako osnovnih, tako i unapređenih za rešavanje problema globalne optimizacije. Algoritmi su testirani na velikom broju benčmark i praktičnih problema. Prikazani rezultati i komparativne analize sa drugim algoritmima svedoče o kvalitetu i efikasnosti metaheuristika rojeva u rešavanju ove klase problema.

Radi lakšeg pregleda, razdvojeni su primeri optimizacije globalnih problema bez i sa ograničenja i praktične primene.

3.4.1 Primeri za globalne probleme bez ograničenja

Kao jedan od najpoznatijih predstavnika metaheuristika rojeva, PSO ima brojne implementacije za globalnu optimizaciju. Tako na primer, PSO sa tri unapređenja, koja poboljšavaju brzinu konvergencije i efikasnost osnovne verzije prikazan je u jednom objavljenom radu [60]. Kao još jedan primer, navodi se PSO verzija sa

dinamičkim podešavanjem vrednosti parametra inercione težine, čime se uspostavlja bolji balans između intenzifikacije i diversifikacije [61].

Jedna od prvih verzija ABC algoritma testirana je na benčmark problemima globalne optimizacije bez ograničenja. U testiranjima su korišćene test funkcije sa velikim brojem varijabli i ABC je pokazao bolje rezultate od GA, PSO i PS-EA (eng. particle swarm inspired evolutionary algorithm) [53].

U objavljenim radovima, takođe se nalaze opisi velikog broj primera CS algoritma. Ova metaheuristika primenjivana je na brojne benčmark probleme globalne optimizacije bez ograničenja [43], i pokazala da ima veliki potencijal u ovom domenu. U jednom radu prikazane su performanse objekt-orjentisanog softverskog okruženja CS algoritma [62]. Algoritam je više puta testiran na standardnim benčmark funkcijama globalne optimizacije bez ograničenja sa različitim brojem varijabli i postignuta su kvalitetna rešenja.

Prva prikazana verzija metaheuristike FA za probleme globalne optimizacije bez ograničenja testirana je na 10 benčmark funkcija sa različitim brojem dimenzija [54]. FA je upoređivan sa performansama GA i PSO metaheuristika, koje su testirane na istom skupu problema, i pokazao se kao efikasnija optimizaciona metoda.

Jedna od retkih implementacija metaheuristike SOA za benčmark probleme je adaptacija algoritma za optimizaciju funkcija realnih varijabli bez ograničenja [63]. Algoritam je pokazao zadovoljavajuće rezultate, što je potvrdila i komparativna analiza sa DE i tri modifikovane verzije PSO algoritma.

KH je takođe prikazan za globalnu optimizaciju bez ograničenja, gde je pokazao da je bolji od drugih 8 dobro poznatih optimizacionih metoda [46]. Slično, u jednom radu prikazane su performanse metaheuristike BA za globalne probleme bez ograničenja. Algoritam je testiran na 10 standardnih benčmark problema i pokazao se uspešnijim nego GA i PSO [58].

Metaheuristika FWA je testirana na 9 problema globalne optimizacije složenih funkcija [59]. Algoritam je upoređen sa dve varijante metaheuristike PSO i postigao je bolju brzinu konvergencije i bolje najbolje rešenje. Kao još neki primeri navode se

unapređeni FWA sa dinamičkim mehanizmom lokalne pretrage [64] i paralelni FWA [65], gde su oba algoritma testirana na problemima bez ograničenja. Paralelna FWA implementacija pokazala se kao bolji pristup od standardnog FWA i PSO, a najbolja rešenja generisala je čak 200 puta brže nego druga dva algoritma.

3.4.2 Primeri za globalne probleme sa ograničenjima

Modifikovani ABC je u jednom objavljenom radu prikazan za probleme globalne optimizacije sa ograničenjima [66]. Testiranja su vršena na standardnom benčmark skupu od 13 funkcija sa različitim brojem evaluacija funkcije cilja. ABC je postigao bolje rezultate od 9 drugih algoritama, a u većini slučajeva i optimalno rešenje benčmark problema. Slično, nadograđeni ABC za probleme inženjerske globalne optimizacije pokazao je zadovoljavajuće performanse [67], [68]. Osim ovih, postoje i drugi primeri ABC algoritma za probleme sa ograničenjima [69].

Slično kao i ABC, i FA je testiran na inženjerskim problemima globalne optimizacije [70] i na drugim benčmark problemima sa ograničenjima [71]. U literaturi se takođe nalazi veliki broj primera, gde je FA adaptiran za rešavanje problema sa više funkcija cilja [72]. CS [73] i BA [74] su takođe testirani na inženjerskim problemima. BA se pokazao kao efikasan algoritam i za rešavanje problema višekriterijumske optimizacije [75].

3.4.3 Praktične primene

Od praktičnih primena PSO algoritma, može da se navede jedna na problem ekonomičnog slanja pošilji [76] i jedna za problem optimizacije portfolija [77]. ABC se takođe pokazao kao efikasna metaheuristika za rešavanje brojnih praktičnih problema [78], [79], kao što je na primer problem raspoređivanja projekata [80].

Od praktičnih primena metaheuristike FA, navodi se jedan primer za problem segmentacije slika [81]. Osim ove, u svetskoj literaturi postoje i mnoge druge aplikacije [82], [83]. CS je primenjivan na probleme poput raspoređivanja poslova [84], dizajniranja sistema [85] i segmentacije slika [81].

SOA je uglavnom testirana na praktičnim problemima. Kao primeri mogu da se navedu primene na problem optimalne distribucije električne energije [86], [87], problem praćenja opterećenja autonomnih sistema električne energije [88] i problem dizajniranja digitalnog filtera [89].

Metaheuristika KH je osim na benčmark problemima testirana i na praktičnim problemima, kao što su problemi geografske optimizacije [90] i optimizacija portfolija hartija od vrednosti [91]. Kao jedna od praktičnih primena FWA metaheursitke, može da se navede primena na problem optimizacije portfolija sa ograničenjima [92].

4 Hibridni algoritmi

Metaheuristike inteligencije rojeva (kao uostalom i bilo koja druga tehnika) nisu idealni optimizatori. Zbog toga postoji stalna potreba za njihovim unapređivanjem i poboljšanjima, o čemu svedoči velika količina stručne literature koja se bavi upravo ovom problematikom. Mnogobrojni radovi u vrhunskim svetskim časopisima posvećeni su analizi nedostataka algoritama rojeva i predlozima kako ovi nedostaci mogu da se isprave.

Algoritmi inteligencije rojeva mogu da se unapređuju na različite načine. Unapređenja metaheuristika rojeva u opštem smislu mogu da budu takva da najvećim delom zadržavaju osnovnu strukturu osnovnog algoritma i menjaju samo pojedine detalje, ili da značajno menjaju ponašanje algoritma uvođenjem bitno novih elemenata.

Prva grupa unapređenja uglavnom se odnosi na modifikaciju jednačina (metoda) pretrage i na optimizaciju pojedinih parametara (lokalnih i/ili globalnih) koji kontrolišu izvršavanje algoritma. Jednačine pretrage se unapređuju na razne načine. Tako na primer, može da se menja vektor (koeficijent) skaliranja koji određuje smer u kome se generiše novo rešenje. Takođe, može da se modifikuje i jačina intenzifikacije, tako što se pri kreiranju novog rešenja menja samo jedna komponenta, više komponenti, ili sve komponente postojećeg rešenja.

Druga grupa unapređenja algoritama rojeva odnosi se pre svega na memetske i hibridizovane algoritme. Memetski algoritmi nastaju inkorporacijom neke od metoda lokalne pretrage u algoritme rojeva. Metode lokalne pretrage brzo pronalaze optimalno rešenje, ali je taj optimum često lokalnog, a ne globalnog karaktera. S druge strane, algoritmi inteligencije rojeva su robusne metaheuristike zato što koriste veliku populaciju jedinki i imaju sofisticirane mehanizme koji omogućavaju izlazak iz lokalnog optimuma. Na ovaj način se kombinacijom algoritama rojeva i metoda lokalne pretrage dobijaju memetski algoritmi koji su s jedne strane brži od "standardnih" algoritama rojeva, a s druge strane bolji u izbegavanju lokalnih optimuma od metoda lokalne pretrage.

Drugi način za implementaciju značajnijih promena u funkcionisanju algoritama

rojeva je mešanje, tj. hibridizacija različitih pristupa. Hibridni algoritam je algoritam koji kombinuje dva ili više algoritama. Neki hibridni algoritmi koriste različite algoritme u zavisnosti od problema koji se optimizuje, dok drugi vrše dinamičko prebacivanje s jednog na drugi algoritam u toku samog izvršavanja.

Postoji više pristupa hibridizaciji. U hibridu niskog nivoa, određena funkcija jednog algoritma (tipično funkcija pretrage) zamenjuje se metodom drugog algoritma. Kod hibrida visokog nivoa, kombinuju se različiti algoritmi inteligencije rojeva, ali bez menjanja načina na koji svaki od pojedinačnih algoritama funkcioniše. Kod hibridnih algoritama koji funkcionisu po principu smene (eng. relay hybridization), algoritmi se primenjuju jedan za drugim, tako što svaki naredni algoritam kao ulaz koristi izlaz prethodnog. Kod hibridnih algoritama koji se baziraju na "timskom radu" svaki algoritam izvodi pretragu nezavisno od ostalih. Takođe, hibridni algoritmi mogu da budu globalni i parcijalni. Kod globalnih hibrida svi algoritmi vrše pretragu celog prostora pretrage, dok se kod parcijalnih hibrida, vrši dekompozicija optimizacionog problema na podprobleme, a zatim svaki algoritam rešava podproblem za koji je zadužen.

Hibridni algoritmi inteligencije rojeva ne nastaju ”slučajnom” kombinacijom pojedinih funkcionalnih elemenata i procedura različitih algoritama, već su oni utemeljeni na sveobuhvatnom izučavanju načina na koji funkcionišu algoritmi koji se hibridizuju. Na taj način se u hibridni algoritam inkorporiraju prednosti jednih, dok se istovremeno eliminišu nedostaci drugih algoritama. Hibridni algoritmi inteligencije rojeva često daju mnogo bolje rezultate od osnovnih implementacija. U literaturi se može naći mnogo radova koji se bave hibridizacijama metaheuristika rojeva.

4.1 Nedostaci metaheurističkih algoritama

Uočena su dva glavna problema algoritama rojeva kod rešavanja problema globalne optimizacije [93]:

1. *Preuranjena konvergencija (eng. premature convergence).* Ovaj problem manifestuje se konvergenciji ka lokalnom optimumu koji nije blizu globalnog op-

timuma. Konačno rešenje može biti dopustivo i da zadovoljava ograničenja prostora pretrage, ali je nezadovoljavajuće i

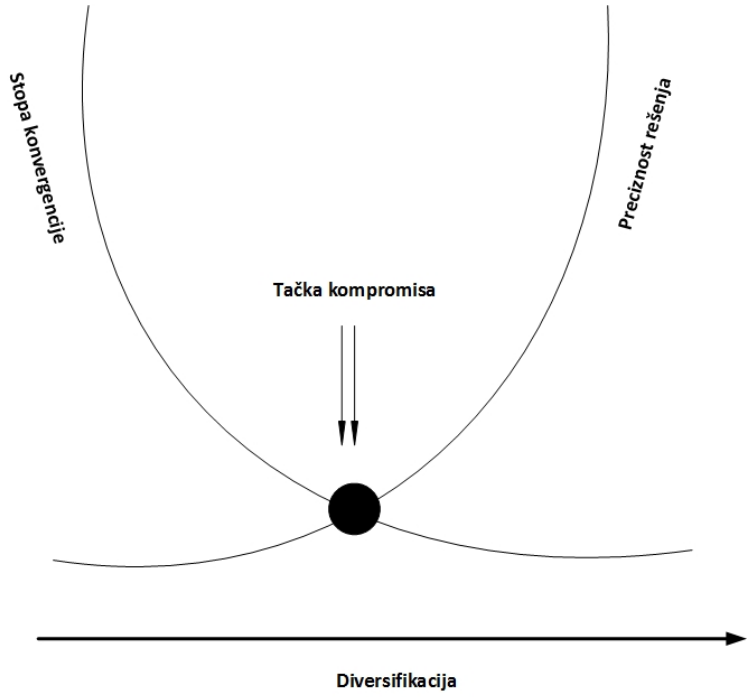
2. *Spora konvergencija (eng. slow convergence).* U procesu optimizacije se kvalitet rešenja ne poboljšava dovoljno brzo. Rešenja ovakvog algoritma jasno pokazuju stagnaciju na grafiku konvergencije (ili u jednoj iteraciji, ili u proseku više iteracija).

Navedeni problemi su najčešće delimično povezani sa diversifikacijom rešenja u populaciji. Diversifikacija se u prirodnim sistemima odražava varijetetom (kvalitetom) i bogatstvom (kvantitetom) organizama na datom prostoru i u vremenu [94].

Diversifikacija jedinki u populaciji je obično visoka na početku procesa pretrage, da bi se zatim smanjivala kako algoritam napreduje prema globalnom optimumu. Velika diversifikacija povećava verovatnoću da će algoritam da pronađe optimalno rešenje, ili bolje suboptimalno rešenje, ali s druge strane i uzrokuje sporu konvergenciju. Dakle, mora postojati balans (kompromis) između konvergencije i kvaliteta rešenja [93]. U suprotnom slučaju, nizak nivo diversifikacije omogućava bržu konvergenciju, ali uz smanjivanje verovatnoće pronalaska optimalnog rešenja. Ovaj scenario je vizualizovan na Slici 4.1. Presek stope konvergencije (eng. convergence rate) i preciznosti naziva se tačka kompromisa (eng. tradeoff point).

Diversifikacija je faktor koji je usko povezan sa širim principima eksploracije i eksploatacije. Visoka diversifikacija pospešuje eksploraciju (istraživanje) prostora pretrage, ali nizak nivo diversifikacije ne znači *a priori* jaču eksploataciju (intenzifikaciju) zbog činjenice da intenzifikacija zahteva poznavanje informacija o kvalitetu rešenja i informacije dobijene iz populacije tokom procesa pretrage. U slučajevima gde je diversifikacija populacije visoka, problem konvergencije je pojačan. Prostim balansiranjem odnosa između eksploatacije i eksploracije može da se utiče da se algoritam ponaša najbolje što može, ali to ne znači da je i brzina (stopa) konvergencije visoka. Zadovoljavajuća konvergencija zahteva pametnu eksploataciju u pravo vreme i na pravom mestu, i ovo je još uvek ostalo kao otvoreno pitanje u literaturi [93].

Osim navedenog, poznatno je da je jedan od glavnih razloga preuranjene konvergencije upravo nedostatak diversifikacije. Zbog toga, da bi algoritam uspeo da se



Slika 4.1: Balans između kvaliteta i stope konvergencije [93]

izbavi iz lokalnog optimuma, određeni nivo diversifikacije mora da se održava tokom izvršavanja algoritama, čak i o trošku sporije konvergencije [93]. Jedan od načina rešavanja ovog problema je hibridizacija, koja je široko prihvaćena kao strategija koja održava diversifikaciju procesa pretrage.

U nastavku su prikazani samo neki od uočenih nedostataka metaheuristika rojeva sa konkretnim primerima.

Jednačine pretrage mnogih metaheuristika rojeva se baziraju na operatoru mutacije, bez korišćenja operatora ukrštanja. Tako na primer, ako se posmatra jednačina pretrage ACO algoritma, vidi se da se slučajna putanja generiše uglavnom pomoću operatora mutacije, dok selekcija na osnovu količine feromona omogućava izbor najkraćih putanja. U ACO pristupu ne postoji eksplicitno definisan operator ukrštanja. Međutim, mutacija se ne vrši samo prostom zamenom vrednosti, kao što je slučaj kod GA, već se nova rešenja generišu pomoću mutacije koja zavisi od podobnosti jedinki. Na primer, neka je verovatnoća da će mrav na grafovskom problemu iz čvora i da izabere čvor j data jednačinom:

$$p_{i,j} = \frac{\phi_{i,j}^{\alpha} d_{i,j}^{\beta}}{\sum_{i,j=1}^{n} \phi_{i,j}^{\alpha} d_{i,j}^{\beta}}, \tag{4.1}$$

gde je $\phi_{i,j}$ koncentracija feromona na putanji između čvora i i j, $d_{i,j}$ atraktivnost putanje, a $\alpha > 0$ i $\beta > 0$ parametri koji određuju uticaj koncentracije feromona i atraktivnosti putanje, respektivno, na verovatnoću izbora. Proces selekcije se u određenoj meri zasniva na *a priori* znanju o određenoj putanji, kao što je dužina putanje $s_{i,j}$, i uzima da je $d_{i,j} \sim 1/s_{i,j}$.

I u slučaju ABC algoritma ne postoji eksplicitna primena operatora ukrštanja, a selekcija se vrši na osnovu podobnosti individua. Neka se za primer uzme osnovna jednačina intenzifikacije ABC algoritma (detaljan opis ABC algoritma je dat u sekciji 5.1.1):

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{4.2}$$

gde je ϕ_{ij} pseudo-slučajan parametar koji kontroliše generisanje novog rešenja u susedstvu rešenja x_{ij} i uzima vrednost iz intervala $[-1, 1]$.

Iz jednačine (4.2) vidi se da se novo rešenje generiše mutiranjem određene varijable trenutnog rešenja, bez primene operatora ukrštanja. Mutacija se vrši u odnosu na vrednost relevantne varijable rešenja iz susedstva.

Dakle, i metaheuristike ACO i ABC koriste mutaciju i selekciju koja zavisi od podobnosti i zato imaju dobar mehanizam globalne pretrage. U opštem slučaju, oba algoritma mogu efikasno da istražuju prostor pretrage, ali uz sporu konvergenciju što je posledica odsustva operatora ukrštanja. Zbog toga je eksploatacija poddomena prostora pretrage jako ograničena.

Kao opšti zaključak izvodi se da oba algoritma pokazuju relativno nisku moć intenzifikacije u odnosu na snagu diversifikacije. Upravo ova činjenica objašnjava zašto oba pristupa pokazuju izvanredne rezultate u rešavanju mnogih teških optimizacionih problema, ali za njihovo rešavanje koriste veliki broj evaluacija funkcija.

Slična pojava dešava se i kod PSO algoritma, kod kojeg su ažuriranje pozicija i brzine čestica date kao:

$$v_i^{t+1} = v_i^t + \alpha\epsilon_1[g^* - x_i^t] + \beta\epsilon_2[x_i^* - x_i^t], \tag{4.3}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{4.4}$$

gde su ϵ_1 i ϵ_2 slučajni vektori između 0 i 1, a α i β su parametri ubrzanja, za koje se obično uzima da je $\alpha = \beta \sim 2$.

Iz jednačina (4.3) i (4.4) vidi se da se pozicije novih rešenja generišu na osnovu mutacije prema obrascima pretrage, dok se proces selekcije vrši implicitno korišćenjem globalno najboljeg rešenja u populaciji g^* i najbolje istorijske pozicije svake jedinke x^*. Međutim, uloga najbolje istorijske pozicije svake individue u populaciji je manje izražena, pošto globalno najbolje rešenje ima daleko veći uticaj na proces selekcije [18].

Dakle, kao i u slučajevima ACO i ABC algoritama rojeva, i PSO koristi uglavnom operatore mutacije i selekcije. Ni PSO ne koristi eksplicitno definisan operator ukrštanja, što implicira da je mobilnost čestica PSO algoritma velika uz jaku eksploraciju. Međutim, korišćenje g^* može da bude "mač sa dve oštrice". Prednost je da kretanje ka trenutno najboljoj poziciji u populaciji može značajno da poboljša brzinu konvergencije, dok u isto vreme može da prouzrokuje preuranjenu konvergenciju ako se g^* ne nalazi u blizini pravog optimuma.

Osim nedostatka operatora ukrštanja, kao još jedna od mana metaheuristika rojeva navodi se da je uočeno da procesi intezifikacije i diversifikacije, kao vodeći mehanizmi pretrage, u nekim slučajevima nisu dovoljno razdvoljeni. Kod mnogih algoritama je nejasno šta je jednačina intenzifikacije, a šta diversifikacije.Takođe, u jednačinama pretrage obično postoji parametar, od koga zavisi veličina koraka pretrage, pa samim tim od vrednosti tog parametra zavisi da li će pretraga biti lokalno (intenzifikacija), ili globalno (diversifikacija) orijentisana.

Tako na primer, ako je vrednost parametra jednačine intenzifikacije ϕ_{ij} ABC algoritma (jednačina 4.2) bliža vrednosti -1 ili 1, pretraga ima globalniji karakter i ako je razlika $x_{ij} - x_{kj}$ velika, nejasno je da li ova jednačina vrši intezifikaciju ili diversifikaciju. Razlika $x_{ij} - x_{kj}$ je veća u početnim iteracijama izvršavanja algoritma kada je diversifikacija između individau velika, a zatim se smanjuje kako algoritam konvergira ka optimalnom rešenju i kako jedinke u populaciji postaju sličnije. Dakle, u početnim fazama izvršavanja, u zavisnosti od generisanog pseudo-slučajnog broja ϕ_{ij}, ova jednačina može da izvodi i proces diversifikacije.

U prilog prethodnoj tvrdnji ide i to da je uočeno da prilikom generisanja novog rešenja jednačinom (4.2), varijable novog rešenja često ispadaju iz granica (pretraga se diversifikuje van granica dopustivog prostora), bilo da se radi o globalnim problemima bez, ili sa ograničenjima. Zato je potrebno primeniti mehanizam za vraćanje vrednosti varijabli u dozvoljeni opseg, što nepotrebno produžava vreme izračunavanja algoritma. U najvećem broju slučajeva, koristi se mehanizam koji vraća vrednosti varijabli na same granice dopustivog opsega. Međutim, ako se ovaj mehanizam koristi često, dešava se da se veliki broj rešenja nalazi na granicama, što utiče na diversifikaciju populacije. Zbog toga se primenjuju i drugi metodi koji vrednosti varijabli vraćaju malo dalje od granica dozvoljenog opsega [95].

Problem metaheuristike ABC koji je upravo naveden pojačava nedostatak koji se ogleda u slabom procesu eksploatacije, što je naročito izraženo u ranijim iteracijama izvršavanja algoritma. Budući da u osnovnom ABC algoritmu pčela izviđač tokom celog izvršavanja izvodi proces diversifikacije, neodgovarajući balans između intenzifikacije i diversifikacije je naročito izražen u ranijim iteracijama, s obzirom da u nekim ciklusima jednačina intenzifikacije više ”vuče” na stranu diversifikacije.

Nasuprotom ovom problemu, ABC pokazuje i nedostatak u kasnijim iteracijama izvršavanja. U ovoj fazi je algoritam pronašao optimalni domen prostora pretrage i tada pčele izviđači nepotrebno troše cikluse algoritma na diversifikaciju, koja više nije potrebna. Dakle, i u kasnijoj fazi izvršavanja balans između eksploatacije i eksploracije nije dobro uspostavljen, pošto bi trebao da bude mnogo više postavljen u korist "intenzifikacije". Povoljna je okolnost da svi navedeni nedostaci mogu da

se isprave hibridizacijom, što je detaljno opisano u Poglavlju 5.

Za razliku od situacije sa ABC algoritmom, FA (ovaj algoritam je detaljno prikazan u sekciji 5.2.1) ne pokazuje nedostatak snage intenzifikacije. FA koristi samo jednu jednačinu pretrage:

$$x_i = x_i + \beta_0 e^{-\gamma r_{i,j}^2}(x_i - x_j) + \alpha\epsilon_i, \tag{4.5}$$

gde parametar slučajnosti α kontroliše veličinu koraka procesa pretrage, a ϵ_i predstavlja vektor pseudo-slučajnih brojeva uniformlno distribuiranih između 0 i 1. Za razliku od parametra ϕ_{ij} ABC algoritma koji je pseudo-slučajne prirode, vrednost parametra α nije slučajna i određuje se na početku izvršavanja algoritma. Sa rastom vrednosti α, pretraga je globalnija i obrnuto. Zbog toga se u početnim ciklusima izvršavanja FA ovaj parametar postavlja na relativno veliku vrednost, a zatim se dinamički menja (smanjuje) u zavisnosti od trenutne iteracije. Na ovaj način, na početku izvršavanja algoritma, kada algoritam još uvek nije konvergirao ka optimalnom domenu, diversifikacija je snažnija, a kasnije se, sa rastom brojača trenutne iteracije, smanjuje.

Empirijski testovi su pokazali da FA jednačina pretrage daje odlične rezultate u slučaju problema globalne optimizacije bez ograničenja. Međutim, za probleme sa ograničenjima, ista jednačina koja izvodi i proces eksploatacije i eksploracije nije dovoljna da bi algoritam postigao dobre rezultate. Uočeno je da kod ovakvih problema, zbog relativno ograničenog prostora pretrage, postoji potreba za jačom diversifikacijom u ranijim ciklusima izvršavanja i da samim tim balans između intezifikacije i diversifikacije nije dobro podešen. Dakle,suprotno od ABC algoritma, FA u početnim fazama izvršavanja uspostavlja balans koji previše „vuče" na stranu intenzifikacije.

Za razliku od metaheuristika ABC i FA, CS uspostavlja odgovarajući balans između intenzifikacije i diversifikacije. Jednačina pretrage algoritma CS enkapsulira lokalni slučajan hod i globalni eksplorativni slučajan hod. Parametar prebacivanja p_a kontroliše koji će od dva mehanizma imati veći uticaj na kretanje individue u populaciji. Lokalni slučajan hod može da se prikaže sledećom jednačinom [18]:

$$x_i^{t+1} = x_i^t + \alpha s \oplus H(p_a - \epsilon) \oplus (x_j^t - x_k^t), \tag{4.6}$$

gde su x_j^t i x_k^t dva različita rešenja iz populacije koja su izabrana slučajnom permutacijom, $H(u)$ je „Heavyside" funkcija, ϵ je pseudo-slučajan broj izvučen iz uniformne distribucije, a s je veličina koraka. Simbol $\oplus$ označava skalarni proizvod.

S druge strane, globalni slučajan hod se izvodi pomoću Lévy letova [18]:

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda), \tag{4.7}$$

gde je [18]:

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) sin(\pi\lambda/2)}{\pi} - \frac{1}{s^{1+\lambda}}, \ (s \gg s_0 > 0) \tag{4.8}$$

U jednačini (4.7), α je faktor skaliranja veličine koraka i zavisi od broja dimenzija problema koji se optimizuje.

Jednačina pretrage CS algoritma ima dve karakteristične prednosti u odnosu na druge algoritme, poput GA i SA: efikasni slučajan hod i balansirana kombinacija lokalne i globalne pretrage. Obzirom da su Lévy letovi jedna od najefikasnijih tehnika slučajne pretrage, CS je jako efikasan u izvođenju globalne pretrage. Nedavna istraživanja su pokazala da CS može da garantuje globalnu konvergenciju [18]. Takođe, u CS algoritmu se na osnovu sličnosti između jaja (individua) proizvode bolja nova rešenja, što u suštini predstavlja generisanje rešenja na osnovu podobnosti. Drugim rečima, CS koristi promenljivu mutaciju koja se izvodi pomoću Lévy letova i mehanizam generisanja novih rešenja na osnovu podobnosti i slučajnosti, što zapravo predstavlja blagu varijantu ukrštanja.

Analizom navedenih jednačina vidi se da CS algoritam dobro balansira intenzifikaciju i diversifikaciju, kao i da ima snažan mehanizam globalne pretrage. Međutim, snaga intenzifikacije CS pretrage je nešto manja, što su pokazali i empirijski testovi. Naime, varijabla trenutnog rešenja se menja u smeru slučajno odabranih rešenja (jednačina 4.6). Ovaj, uslovno rečeno, nedostatak posebno dolazi do izražaja kada

parametar α ima veću vrednost i u tom slučaju lokalna pretraga se više ponaša kao globalna, tj. diversifikacija. Kao implikacija navedenog, CS ima nešto slabiju brzinu konvergencije i u tom domenu ima prostora za poboljšanja. Takođe, pretraga previše zavisi od vrednosti parametara, posebno parametra α, tako da je za svaki problem potrebno posebno, empirijskim putem, izvršiti podešavanja parametara, tj. ne postoji skup parametara pomoću kojeg će se ostvariti zadovoljavajući rezultati za širi set benčmark problema. Navedeni nedostaci se najbolje vide kada se algoritam koristi za optimizacione probleme sa ograničenjima. U slučaju globalnih problema bez ograničenja, CS postiže odlične rezultate.

Za razliku od algoritma CS, metaheuristika FWA nema kontrolu nad balansom između eksploatacije i eksploracije. FWA za izvođenje procesa intenzifikacije koristi jednačinu (4.9) na osnovu koje se određuje broj varijabli problema koje će biti izložene promenama u procesu pretrage. Ovaj algoritam detaljno je opisan u Sekciji 5.4.1.

$$z = round(d \cdot \chi), \tag{4.9}$$

gde d predstavlja broj varijabli problema koji se optimizuje, dok je χ psudo-slučajan broj uniformno distribuiran između 0 i 1. Dakle, od vrednosti pseudo-slučajnog broja χ zavisi da li će pretraga biti više lokalno (ako je ovaj parametar manji) ili globalno (ako je ovaj parametar veći) orjentisana. Na ovaj način ne postoji eksplicitna kontrola nad balansom između eksploatacije i eksploracije, niti je jasno kada i u kojoj meri jednačina pretrage izvodi intenzifikaciju, a kada diversifikaciju. Empirijskim putem je utvrđeno da na ovaj način algoritam konvergira previše sporo ka optimumu i da je potrebno da se pojača snaga intenzifikacije, posebno u početnim fazama izvršavanja.

U Glavi 5, gde su prikazani hibridni algoritmi koji su implementirani, detaljno su prikazani nedostaci svakog algoritma koji je hibridizovan.

4.2 Memetski algoritmi

Postoji tendencija sve češćeg ugrađivanja tehnika lokalne pretrage u metode pretrage evolutivnih algoritama. Ovakvi algoritmi su poznati u literaturi pod raznim nazivima, kao što su memetski algoritmi (eng. memetic algorithms – MA), evolutivne lokalne pretrage (eng. evolutionary local search), genetske lokalne pretrage (eng. genetic local search) i hibridi globalne-lokalne pretrage (eng. global-local search hybrids). Pojam memetski algoritmi prvi je upotrebio Pablo Moscato 1989. godine [96], dok je termin memetski izveo biolog Charles Darwin na osnovu teorije o „memama", kao kulturološkim i sociološkim ekvivalentima gena.

U ranim fazama razvoja, memetski algoritmi su bili modifikacije genetskih algoritama koje koriste osim standardnih genetski operatora i operator lokalne pretrage. Nova, vizionarska metafora memetskih algoritama data je u vreme kada su hibridni optimizacioni algoritmi već bili u upotrebi [96]. U početku je naučna zajednica posmatrala memetski pristup s nevericom, a tek nakon deset godina od pojavljivanja memetskih algoritama počelo je masovno objavljivanje radova iz ove oblasti.

Memetski algoritmi predstavljaju kompleksne strukture, kao što je kombinacija prostih agenata ili mema čije interakcije evolutivnog karaktera vode do inteligentnih ponašanja sposobnih da rešavaju brojne probleme [97]. Prema još jednoj definici, memetski algoritmi su populacione metaheuristike koje se sastoje iz evolutivnog okvira i skupa algoritama lokalne pretrage koji se pozivaju tokom ciklusa izvršavanja eksternog okvira [97].

Razvoj modernih tehnika koje su inspirisane kulturološkim mešanjima, ali pripadaju definici memetskih metoda sugeriše koncept memetskog računarstva (eng. memetic computing). Ovo je širi koncept definisan u [98] kao paradigma koja koristi ideju mema kao jedinice informacija predstavljene kroz računarsku reprezentaciju sa svrhom da se reše određeni problem. Drugim rečima, deo istraživača iz oblasti računarstva je pokušao da proširi koncept meme u kontekstu rešavanja problema na nešto šire i inovativno [99]. Činjenica da *ad hoc* optimizacioni algoritmi (algoritmi koji koriste znanje o specifičnom problemu) mogu efikasno da rešavaju optimizaci-

one probleme argumentovana je rezultatima iz literature. S druge strane, krajnji cilj svake metaheuristike je generisanje autonomnih i inteligentnih struktura [97]. Cilj optimizacije pomoću računarske inteligencije je automatizovani dizajn optimizacionih algoritama u realnom vremenu. Memetsko računarstvo može da se posmatra i kao oblast koja proučava kompleksne strukture koje su sastavljene od jednostavnih modula (mema) koje interaguju i evoluiraju prilagođavajući se problemu kojeg rešavaju.

Ovakav pogled na ovu temu vodi ka modernijoj definiciji memetskog računarstva datoj u [100]. Memetsko računarstvo je širi pojam koji se bavi složenim i dinamičkim računarskim strukturama koje se sastoje iz modula (mema) koje interaguju jedna sa drugom i koje prate evolutivnu dinamiku koja je inspirisana idejama o mešanju. Meme su jednostavne strategije čije visoko-usaglašene akcije omogućavaju rešavanje različitih problema. Da bi se podvukla jasnija granica između memetskih algoritama i memetskog računarstva, memetski algoritmi mogu da se definišu kao klasa algoritama koja ima specifične karakteristike, kao što je na primer populacija potencijalnih rešenja, struktura generacija i lokalna potraga u okviru generacije [100]. S druge strane, memetsko računarstvo je oblast koja se bavi proučavanjem struktura algoritama koji se sastoje iz više operatora. U ovom svetlu, memetski algoritmi su kamelj temeljac i osnivački podskup memetskog računarstva. Glavna razlika između ova dva koncepta je algoritamska filozofija koja stoji iza njih. Dok memetski algoritmi spadaju u grupu optimizacionih metoda, memetsko računarstvo je pristup usko vezan za skup operatora bez unapred definisane strukture, ali sa jasno definisanim ciljem rešavanja problema [100].

Kao sto je već navedeno, memetski algoritmi inkorporiraju lokalnu pretragu kako bi se brzo pronašle visoko podobne individue u populaciji i otkrili regioni prostora pretrage koji „obećavaju". U literaturi postoji veliki broj radova gde su prikazani eksperimenti koji nedvosmisleno pokazuju da ovaj pristup daje bolje rezultate nego klasični evolutivni algoritmi.

Prednosti lokalne pretrage su višestruke. Kao prvo, veća je verovatnoća da će brže biti pronađeno rešenje koje ima veliku podobnost. Tokom izvršavanja algoritma,

često se dešava da se pronađe rešenje koje ima malu podobnost, a nalazi se u okolini lokalnog optimuma. U standardnim evolutivnim algoritmima ovakva rešenja se sa velikom verovatnoćom gube tokom procesa selekcije. Međutim, memetski algoritmi mogu da poboljšaju ovakva rešenja i da dostignu bolji lokalni optimum. Ovaj efekat je posebno vidljiv kod optimizacije sa ograničenjima, gde se u velikom broju slučajeva nedopustiva rešenja kažnjavaju u meri koja je proporcionalna sa njihovom podobnošću, pa se kazna smanjuje prema dopustivom regionu. Drugo, ako se primenom operatora mutacije i/ili ukrštanja generišu nedopustiva rešenja, metoda lokalne pretrage se ponaša kao mehanizam popravke koji pronalazi dopustiva rešenja. Konačno, moguće je da se uključi i specifično znanje o problemu u metodu lokalne pretrage. Ova mogućnost je izvesna, pošto je relativno lako da se dizajnira strategija lokalne pretrage, čak i u slučajevima gde ne postoji globalna strategija koja je specifična za određeni problem.

Jedan od ekstremnih primera memetskih algoritama je iterativna lokalna pretraga (eng. iterated local search) [101]. U ovoj implementaciji se kod svakog novokreiranog potomka primenjuje metoda lokalne pretrage sve dok se ne pronađe lokalni optimum. Iz ovoga sledi da se populacija sastoji uvek iz lokalnih optimuma, i algoritmu je potrebna jaka mutacija, koja se naziva peturbacija, da bi se izbavio iz ovakve situacije.

Proces izvršavanja memetskih algoritama svodi se na sledećih četiri koraka [97]:

1. *selekcija roditelja za sledeću generaciju.* Ovaj proces ima za cilj da izabere kandidat rešenja koja će služiti za kreiranje novih rešenja u sledećoj iteraciji izvršavanja algoritma. Kriterijumi za selekciju se najčešće baziraju na podobnosti kandidat rešenja koja neposredno zavisi od vrednosti funkcije cilje f_m koja je predmet optimizacije. Kod memetskih algoritama primenjuje se više metoda selekcije, kao što je selekcija ruletom. Takođe se koristi i metod koji uzima u obzir diversifikaciju rešenja. U ovom slučaju samo dovoljno diversifikovana rešenja preživljavaju i propagiraju se u sledeću generaciju. Ako su rešenja u populaciji dovoljino diversifikovana, proces selekcije može biti i slučajan.

2. *kombinovanje roditelja koji učestvuju u kreiranju potomaka.* Ovim procesom se kreiraju nova, obećavajuća kandidat rešenja kombinovanjem pojedinih strukturnih elemenata prethodnih rešenja.
3. *lokalna poboljšavanja novokreiranih rešenja.* Cilj ovog procesa je poboljšavanje rešenja potomka u što većoj meri. Ova rešenja prolaze kroz proces usavršavanja, što odgovara paradigmi učenja tokom celog života.
4. *ažuriranje populacije.* U ovom koraku donosi se odluka da li da novokreirano rešenje bude deo populacije i da zameni prethodno rešenje. Ova odluka se uglavnom donosi na osnovu kvaliteta i diversifikacije.

Pregledom literature vidi se su memetki algoritmi poslednjih godina mnogo puta primenjivani na rešavanje kompleksnih problema iz realnog života i da postižu dobre performanse za probleme velikih dimenzija. Tako na primer, u [102] i [102] prikazana je memetska optimizacija inženjerskih problema. Rešavanje problema dizajna digitalnog filtera putem memetske metaheuristike inspirisane algoritmom DE je prikazano u [103], dok je memetska varijanta R-NSGA-II algoritma za problem određivanje referentnih tačaka data u [104]. Memetski algoritam sa ciljem da pronađe pareto optimum uz zadovoljavanje kriterijuma preciznosti i diversifikacije predložen je u [105]. Kao neki od novijih primera navodi se primer selekcije opsega za višespektralne slike korišćenjem probabilističkog memetskog algoritma [106], primena memetskih samoadaptirajućih evolutivnih strategija na problem maksimalne diversifikacije [107], primer za upravljanje projektima u uslovima ograničenih resursa [108], kao i optimizacija lokacije postrojenja pomoću paralelnog memetskog algoritma [109].

4.3 Hibridni algoritmi

Najbolje rezultate u optimizaciji mnogih praktičnih i benčmark problema postižu hibridni algoritmi. Zbog toga je interes istraživača za optimizaciju pomoću hibridnih metaheuristika značajno porastao prethodnih nekoliko godina.

Hibridni algoritam je algoritam koji se sastoji iz dva ili više algoritama koji se izvršavaju zajedno i komplementarni su jedan drugom, tako da proizvode sinergijski

efekat [110]. Kod ovih algoritama, dva ili više algoritama sarađuju u rešavanju unapred definisanog i određenog problema. Hibridizacija evolutivnih algoritama je popularna tehnika delimično i zbog činjenice da ovakvi algoritmi postižu bolje performanse u upravljanju šumovima, neizvesnostima i nepreciznostima [111].

Hibridni algoritmi kombinuju različite pristupe na „pametan" način, tako što se nedostaci jednog algoritma eliminišu prednostima drugog. To znači da kreiranju hibridnog algoritma prethodni proces kritičke analize prednosti i nedostataka algoritama koji se kombinuju. Potrebno je pažljivo analizirati strukturu svakog algoritma i njihove osnovne komponente, kao što su jednačine intenzifikacije i diversifikacije, njihov balans, parametri i brzina konvergencije.

4.3.1 Taksonomija hibridnih algoritama

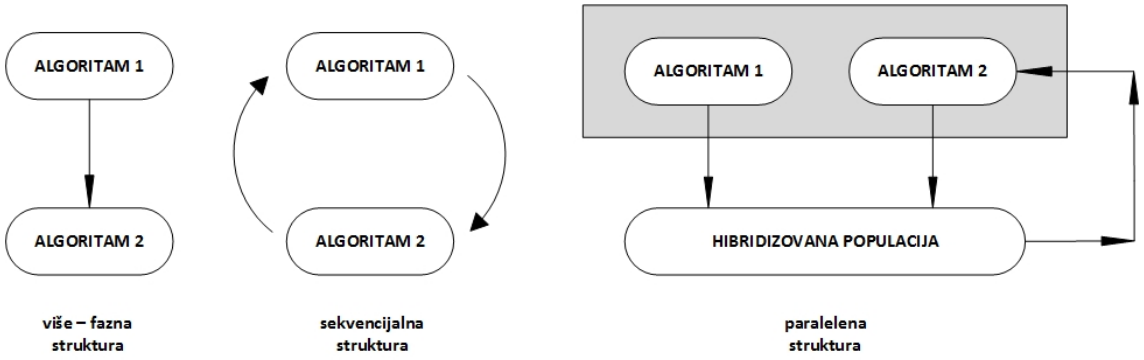
Glavni cilj taksonomije je da se upostavi mehanizam koji omogućava kvalitativno poređenje hibridnih algoritama. U literaturi je poznat veliki broj modela kategorizacije hibridnih algoritama i veliki broj različitih vrsta hibrida. Tako na primer, poznati su hibridi, gde se jedan algoritam ugrađuje u drugi u formi podalgoritma i kao takav služi za određivanje optimalnih parametara algoritma u koji je ugrađen. Kod druge vrste, u jedan algoritam se ugrađuju strukturalni elementi drugog algoritma, kao što su jednača pretrage i mehanizmi ukrštanja i mutacije. S obzirom na navedeno, prema jednoj podeli, hibridni algoritmi u najopštijem smislu mogu da budu [93]:

1. *Hibridi sa ujednačenom svrhom (eng. unified purpose hybrids).* Kod ove vrste hibrida, svi podalgoritmi rešavaju isti problem, ali se uključuju u različitim fazama proesa pretrage. Tipičan predstavnik ove grupe je metaheuristika sa ugrađenim mehanizmom lokalne pretrage, gde se globalnom pretragom ispituje domen potencijalnih rešenja, a lokalna pretraga se koristi za fino istraživanje prostora u kome se može naći globalni optimum.
2. *Hibridi sa više svrha (eng. multiple purpose hybrids).* U ovom slučaju samo glavni algoritam neposredno rešava problem, dok se podalgoritmi koriste za fino podešavanje parametara glavnog algoritma. Tako na primer, ABC al-

goritam može da se koristi za određivanje vrednosti parametara ukrštanja i mutacije GA koji vrši optimizaciju problema. U ovu grupu spadaju i tzv. hiper-heuristički algoritmi (eng. hyper-heuristic algorithms), gde se vrednosti parametara glavnog algoritma određuju pomoću podalgoritma ili nekog mehanizma učenja [112].

Prema drugoj podeli, hibridne metaheuristike se dele na kolaborativne (eng. collaborative hybrids) i na integrativne (eng. integrative hybrids) [112].

Kolaborativni hibridi uključuju dva ili više algoritama koji vrše optimizaciju na sekvencijalan ili paralelan način. U najprostijem slučaju obim optimizacije se ravnomerno dodeljuje svakom algoritmu. Ovi algoritmi mogu imati tri strukture, koje su prikazane na Slici 4.2.



Slika 4.2: Strukture kolaborativnih hibridnih algoritama

Prva struktura se naziva više-fazna (eng. multi-stage). U ovom slučaju postoje dve faze izvršavanja. Prvi algortam vrši globalnu optimizaciju, dok drugi izvodi lokalnu pretragu. Više-fazna struktura se poklapa sa okvirom strategije orlova (eng. eagle strategy), čiji je opis dao Yang [113]. Prvi algoritam, koji izvodi globalnu pretragu, poseduje prednosti koje omogućavaju detektovanje obećavajućih regiona prostora pretrage. Nakon lociranja domena gde bi mogao da se nalazi globalni optimum, drugi algoritam izvodi intenzivnu lokalnu pretragu pomoću tehnika kao što su metoda penjanja uz brdo (eng. hill-climbing) ili simpleks metoda silaska sa brda (eng. downhill simplex method) [114].

Jedan od glavnih izazova više-faznih hibrida jeste određivanja vremenskog trenutka u izvršavanju algoritma kada je potrebno pretragu preusmeriti na drugi algoritam.

Kao jedan od načina rešavanja ovog problema navodi se korišćenje mere diversifikacije [93]. Neki od radova se bave korišćenjem GA kao globalnog optimizera (prvi algoritam) i PSO algoritma kao algoritma koji izvodi lokalnu pretragu (drugi algoritam) [115].

Kod *sekvencijalne strukture (eng. sequential structure)* algoritmi se smenjuju u izvršavanju sve dok jedan od kriterijuma konvergencije ne bude zadovoljen. Radi jednostavnosti implementacije, hibridni algoritam se podesi na način da se svaki algoritam izvršava određeni broj iteracija pre nego što se proces pretrage preusmeri na drugi algoritam.

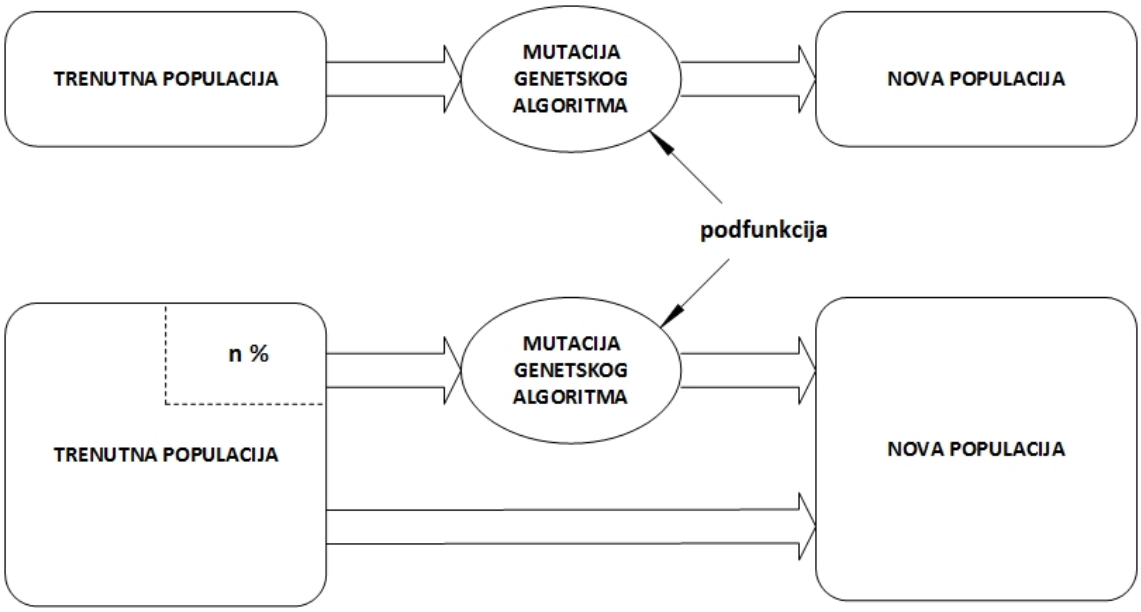
U *paralelnoj strukturi (eng. parallel structure)* algoritmi se simultano izvšavaju nad istom populacijom potencijalnih rešenja.

Sa aspekta *integrativnih hibrida*, algoritam poznat pod nazivom podređeni (eng. subordinate) se ugrađuje u glavni (eng. master) algoritam, pri čemu podređeni algoritam učestvuje sa relativno malim procentom u procesu pretrage [93]. Ova vrsta hibrida se najčešće implementira tako što se operator pretrage sekundarnog algoritma ugrađuje u glavni algortitam. Tako na primer, u literaturi postoji više algoritama kod kojih se operator mutacije iz GA ugrađuje u PSO [116]. Takođe, u nekim pristupima, gradijentne tehnike poput penjanja uz brdo, najstrmijeg spuštanja i Newton-Raphson se ugrađuju u primarni algoritam [117].

Pristup integrativnih hibrida može da se realizuje na dva načina [93]:

1. *Potpuna manipulacija (eng. full manipulation.)* Kod ovih algoritama cela populacija se pretražuje u svakoj iteraciji. Ova operacija se najčešće ugrađuje u izvorni kod kao podfunkcija.
2. *Parcijalna manipulacija (eng. partial manipulation).* U ovom slučaju se samo pretraga u delu populacije ubrzava nekom od tehnika lokalne pretrage. Najveći izazov predstavlja izbor pravog dela domena pretrage i selekcija pravog kandidat rešenja čija će okolina da se istražuje primenom lokalne pretrage.

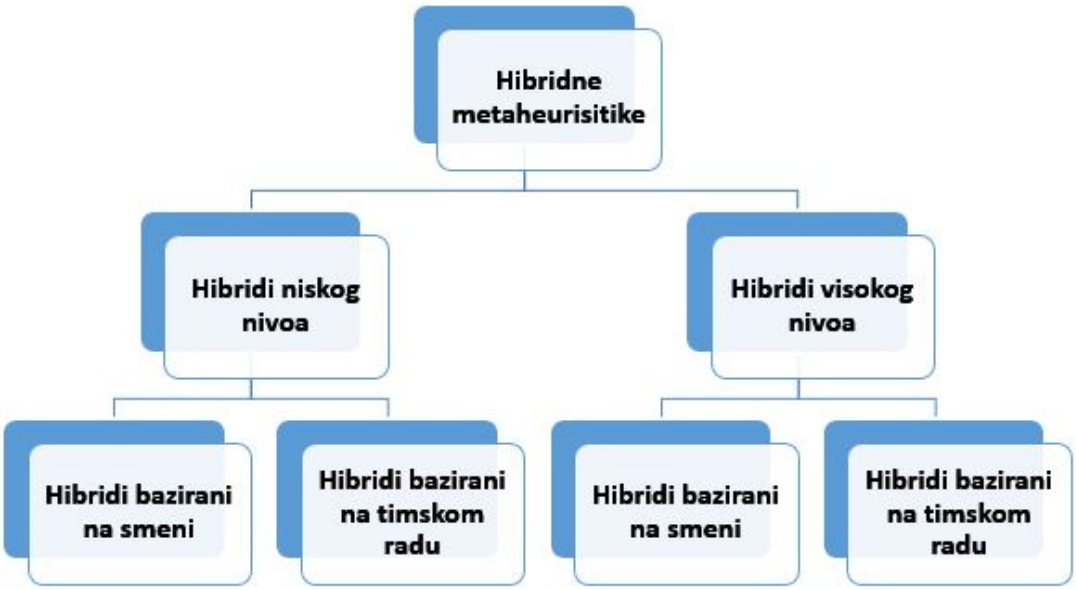
Pristupi potpune i delimične manipulacije prikazani su na Slici 4.3.



Slika 4.3: Integrativni hibridni algoritmi sa putpunom i delimičnom manipulacijom

Talbi je u [118] napravio razliku između hijerarhijske (eng. hierarchical) i ravne (eng. flat) kategorizacije hibridnih pristupa.

Vrste hibridnih algoritama prema hijerarhijskoj klasifikaciji prikazane su na Slici 4.4. Sa slike se vidi da se hijerarhijska klasifikacija izvodi na dva nivoa. Na prvom nivou kategorizacije pravi se razlika između hibrida niskog nivoa (eng. low-level hybrids) i hibrida visokog nivoa (eng. high-level hybrids). Hibridi niskog nivoa se odnose na funkcionalnu kompoziciju jednog optimizacionog metoda. Kod ove klase algortiama, funkcija pretrage jedne metaheuristike se zamenjuje sa funkcijom druge. S druge strane, kod hibrida visokog nivoa kombinuje se više različitih metaheuristika tako što se ne menjaju unutrašnje procedure pojedinih algoritama. Drugi nivo kategorizacije pravi razliku između algoritama baziranih na principu smene (eng. relay hibridiyzation) i algoritama baziranih na timskom radu (eng. teamwork hybridization). Kod hibridizacije na osnovu timskog rada, metaheuristike se primenjuju jedna za drugom, tako što izlaz iz prethodnog algoritma usmerava kao ulaz u sledeći. Dakle, ovaj vid hibrida funkcioniše poput cevovoda. Hibridizacija bazirana na timskom radu predstavlja kooperativne optimizacione modele, gde mnogo agenata paralelno pretražuju domen pretrage [118].



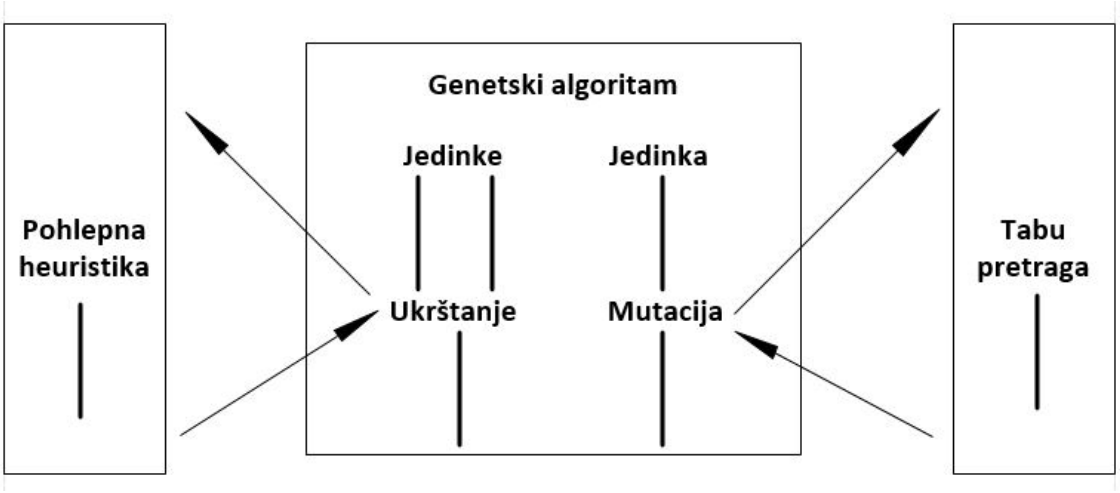
Slika 4.4: Hijerarhijska klasifikacija hibridnih algoritama [118]

Ako se uzme u obzir i prvi i drugi nivo mogu da se izvedu četiri klase hibridnih algoritama. Prva klasa predstavlja hibride niskog nivoa koji funkcionišu po principu smene (eng. low-level relay hibridization - LRH). U ovoj klasi se jedna metaheuristika ugrađuje u drugu metaheuristiku. U literaturi postoji nekoliko primera ove klase.

Klasa hibrida niskog nivoa koji funkcionišu po principu timskog rada (eng. low-level teamwork hybrid - LTH) je struktuirana sa ciljem da se ostvare dva oprečna, ali jednako važna cilja - intenzifikacija i diversifikacija. Diversifikacija je potrebna da bi se osiguralo da je svaki deo prostora pretrage dovoljno ispitan i da bi se dobila pouzdana procena o tome gde se nalazi globalni optimum, dok se intenzifikacija koristi za finu pretragu oko postojećih rešenja u populaciji. Prema jednom stanovištu, populacione heuristike poput genetskih algoritama, razbacane pretrage, mravljih kolonija, itd. dobro izvode diversifikaciju, dok su loše u intenzifikaciji [118].

Zbog navedenog, najefikasnije populacione metaheuristike se kombinuju sa heuristikama lokalne pretrage poput metoda penjanje uz brdo, simuliranog kaljenja i tabu pretrage koje su uspešne u procesu intenzifikacije. Ove dve klase algoritama imaju komplementarne snage i slabosti. Metode lokalne pretrage vrše lokalnu optimizaciju, dok populacione metaheuristike pretražuju prostor globalno [118]. Dakle, kod

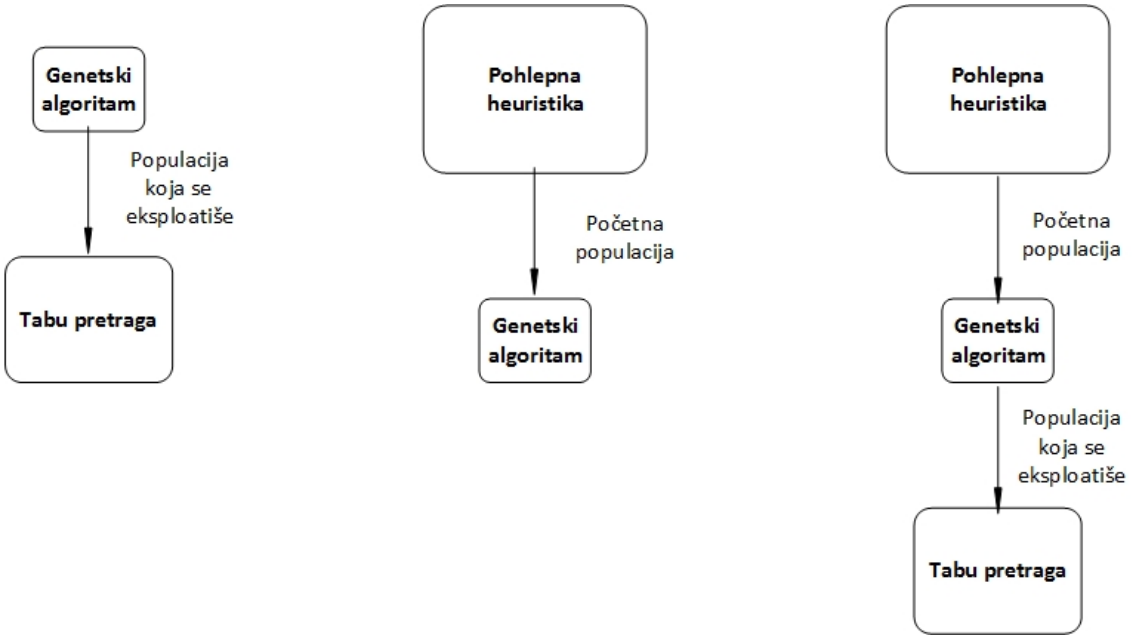
hibrida niskog nivoa baziranih na principu timskog rada, metaheuristički pristupi se najčešće ugrađuju u populacione metaheuristike. Kao jedan primer navodi se genetski algoritam, gde operaciju mutacije izvodi metoda tabu pretrage, a operaciju ukrštanja izvodi metod gramzive selekcije. Ovde se vidi da u literaturi ne postoji jasno definisana razlika između memetskih i hibridnih algoritama. Navedeni primer je dat na Slici 4.5.



Slika 4.5: Primer hibrida niskog nivoa po principu timskog rada [118]

Hibrid visokog nivoa koji funkcioniše po principu smene (eng. high-level relay hibrid - HRH) predstavlja kombinaciju samostalnih metaheuristika koje se izvršavaju jedna za drugom [118]. Tako na primer, često se kombinuju evolutivni algoritmi, koji veoma dobro pronalaze regione visokih performansi kompleksnog prostora pretrage, sa heuristikama koje vrše finu pretragu u okolini optimalnih rešenja. U praksi nakon određenog vremena izvršavanja algoritma populacija postaje uniformna i podobnost jedinki se ne povećava. U takvim situacijama, izgledi da će algoritam otkriti podobnije individue su vrlo skromni i proces pretrage se verovatno zaglavio u nekom od lokalnih regiona, sa malim izgledom da će da se izvuče. U ovakvim situacijama je poželjna intenzivna eksploatacija u okviru već pronađenih regiona koja se bolje izvodi primenom neke od heurističkih metoda. Primer hibrida visokog nivoa koji funkcioniše po principu smene dat je na Slici 4.6.

Hibrid visokog nivoa baziran na timskom radu (eng. high-level teamwork hybrid - HTH) obuhvata nekoliko samostalnih algoritama koji paralelno i kooperativno iz-



Slika 4.6: Primer hibrida visokog nivoa baziran na smeni [118]

vode pretragu. Po logici stvari, ovaj hibrid bi u najmanju ruku trebao da bude efikasan kao jedan od obuhvaćenih algoritama. U praksi se pokazalo da ovi algoritmi rade bolje nego samostalni algoritmi zato što algoritmi jedan drugome pružaju informacije i time pospešuju pretragu [118].

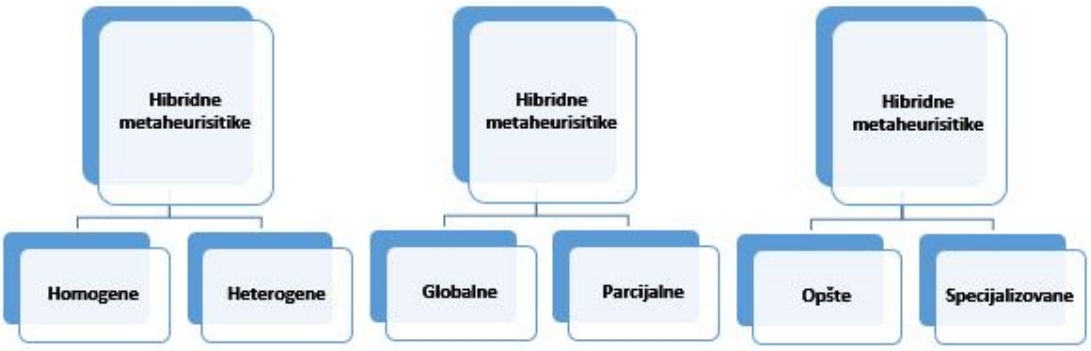
Kao jedan od primera ove klase hibrida navodi se model ostrva genetskog algoritma (eng. GAs island model) [118]. Kod ovog modela, cela populacija se deli na manje podpopulacije na osnovu geografske lokacije. Proces pretrage se kontroliše pomoću nekoliko parametara: topologije koja definiše konekcije i veze između podpopulacija, stope migracije koja određuje broj individua koje prelaze iz jedne u drugu podpopulaciju, strategije zamene individua i migracioni interval koji određuje koliko često se migracija izvodi.

Druga vrsta taksonomije koju je predložio Talbi [118] naziva se ravna klasifikacija (eng. flat classification). Ova kategorizacija prikazana je na Slici 4.7.

Prema ravnoj klasifikaciji [118], prva podela hibridnih algoritama je na homogene i heterogene. U slučaju homogenih hibrida, svi kombinovani algoritmi koriste istu metaheuristiku. Uglavnom se koriste posebni parametri za svaki algoritam. Primer ove vrste je model genetskog algoritma sa ostrvima. S druge strane, kod heterogenih

hibrida kombinuju se različiti algoritmi.

S druge tačke gledišta, hibridni algoritmi mogu biti globalni i parcijalni [118]. Kod globalnih hibrida, svi algoritmi pretražuju ceo domen pretrage. Cilj je da se prostor pretraži što temeljnije. Svi nabrojani hibridi po hijerarhijskoj klasifikaciji su globalni zato što svi algoritmi rešavaju ceo optimizacioni problem. Kod parcijalnih hibridnih pristupa, prvo se vrši dekompozicija problema na podprobleme i svaki algoritam rešava svoj podproblem. U ovom pristupu su uglavnom svi podproblemi međusobno povezani i uključuju ograničenja između optimuma koji pronalazi svaki od algoritama. Iz tog razloga algoritmi komuniciraju kako bi poštovani ova ograničenja i gradili globalno dopustivo rešenje problema.



Slika 4.7: Ravna klasifikacija hibridnih algoritama

Konačno, prema poslednjoj podeli, hibridni algoritmi mogu biti opšti i specijalizovani. Opšti hibridi rešavaju isti optimizacioni problem. Specijalizovani hibridi kombinuju algoritme, gde svaki algoritam rešava zaseban problem. Svi gore navedeni algoritmi pripadaju opštoj kategoriji.

4.3.2 Pregled razvoja hibridnih algoritama

Evolutivni algoritmi, evolutivno programiranje [119], evolucione strategije [120] i genetski algoritmi [121]), kao stohastičke globalne optimizaione metode koje emuliraju proces biološke evolucije, bile su među prvima koje efikasno lociraju globalni optimum u velikom prostoru pretrage, a posebno u situacijama kada za dati problem ne postoji dovoljno informacija koje bi pomogle u njegovom rešavanju. Implementacije

ovih algoritama su jednostavne pošto se oni temelje na relativno prostim konceptima. Oni su takođe fleksibilni, pošto se njihovi parametri lako mogu menjati u cilju postizanja boljih performansi [93].

U literaturi iz ranijeg perioda mogu da se nađu mnoge hibridne implementacije raznih evolutivnih algoritama, ali u svima njima i dalje postoje dva osnovna problema - spora i preuranjena konvergencija. Osim toga, ovi hibridi su previše "skupi" zato što rešavanje složenih problema zahteva veliki broj iteracija.

Oblast hibridnih algoritama je živa i dinamična oblast, gde mnogi istraživači permanentno ulažu velike napore i trud, o čemu svedoče brojni objavljeni radovi. Posebno su u poslednjih nekoliko decenija opažena značajna unapređenja hibridnih algoritama koja su utemeljena na dugogodišnjem istraživanju. Tako na primer, Rodriguez i ostali [122] su implementirali algoritam koji koristi evolutivni algoritam sa simuliranim kaljenjem.

Hibridni pristupi iz literature se ukratko mogu klasifikovati kao [111]:

1. Hibridizacija jednog evolutivnog algoritma sa drugim (na primer genetski algoritam sa genetskim programiranjem);
2. Hibridizacije evolutivnih algoritama i algoritama inteligencije rojeva;
3. Hibridizacija jednog algoritma roja sa drugim algoritmom roja;
4. Hibridizacija neuronskih mreža i evolutivnih algoritama/algoritama rojeva;
5. Hibridizacija metode fazi logike i evolutivnih algoritama/algoritama rojeva i
6. Hibridizacija evolutivnog algoritma/algoritma roja sa drugom heuristikom, kao što su metode lokalne i tabu pretrage, simuliranog kaljenja, penjanje uz brdo i slično.

U navedenoj podeli, iako su algoritmi inteligencije rojeva podskup evolutivnih algoritama, radi boljeg pregleda, podvučena je razlika između ova dva koncepta. U ovom delu, prikazane su hibridizacije evolutivnih algoritama, pošto je detaljan pregled hibridizacija algoritama rojeva dat kasnije.

Tan i drugi [123] su prikazali hibridnu evolutivnu klasifikacionu tehniku za određivanje pravila klasifikacije koja se koriste u medicinske svrhe radi boljeg razumevanja i

prevencije neželjenih događaja. U prvoj fazi izvršavanja određuju se granice prostora pretrage tako što se evoluira populacija dobrih kandidat rešenja. Genetsko programiranje se koristi za razvoj uslovnih (nominalnih) atributa za slobodno struktuirana pravila, dok GA vrši optimizaciju numeričkih atributa za preciznu klasifikaciju pravila bez potrebe za diskretizacijom (podelom na konačne elemente). Na ovaj način generisana kandidat pravila se potom koriste u drugoj fazi u kojoj se optimizuje broj i stepen prvila evolutivnim procesom, čime se formiraju precizni i sveobuhvatni skupovi pravila. Hibridna evolutivna učeća šema za sintezu višeklasnih obrazaca sistema za prepoznavanje razvili su Zmuda i drugi [124]. U prikazanom radu, autori su razvili složene atribute koji se koriste kao ulazi u prosti mehanizam klasifikacije. Skup nelinearnih atributa se kreira kombinacijom genetskog programiranja za sitenzu aritmetičkih izraza i GA koji služi za izbor dopustivog skupa izraza. Krajnji cilj je kreiranje kompaktnog skupa nelinearnih atributa koji učestvuju u rešavanju višeklasnog problema za prepoznavanje obrazaca.

U literaturi se takođe pronalaze brojni primeri hibrida evolutivnih algoritama i neuronskih mreža. Tako je Wang [125] predložio hibridnu metaheuristiku koja poboljšava performanse evolutivnih algoritama za simulaciju optimizacionog problema [111]. U prvoj fazi izvršavanja algoritma konstruiše se neuronska mreža na osnovu trening podataka. Nakon toga koristi se evolutivni algoritam koji generiše nova rešenja na osnovu istraživanja prostora pretrage. Neuronske mreže se zatim koriste za određivanje vrednosti funkcije podobnosti novokreiranog rešenja, kako bi evolutivni algoritam mogao da nastavi sa procesom pretrage. U cilju unapređenja performansi i robusnosti hibridnog algoritma, predloženo je da se koristi više neuronskih mreža [125].

Herrera je predložio adaptivni pristup u kome je problem preuranjene konvergencije i stagnacije GA rešen primenom fazi logičkih kontrolera [126]. Fazi logički kontroleri se koriste za određivanje parametra verovatnoće mutacije GA. Na osnovu empirijskih rezultata prikazanih u radu, zaključuje se da je ovaj hibrid poboljšao osnovni GA.

Evolutivni algoritmi su hibridizovane i sa drugim metodama. Tako na primer, u [127], prikazana je hibridizacija PSO algoritma sa modifikovanom Broyden-Fletcher-

Goldfarb-Shanno (BFGS) metodom koja ima za cilj da poboljša lokalnu pretragu osnovnog PSO algoritma. Algoritam je testiran na dvadeset problema sa više optimuma. Takođe, u literaturi se nalazi i kombinacija GA sa Taguchi metodom [128] za globalnu optimizaciju bez ograničenja. U pitanju je hibrid niskog nivoa u kome Taguchi metoda pomaže operatoru ukrštanja GA u izboru boljih gena. Algoritam je testiran na petnaest standardnih benčmark problema sa 30 i 100 dimenzija. Ostvareni su značajni rezultati.

4.3.3 Nedostaci i izazovi hibridnih algoritama

Yang je u svom radu [93] napravio kratak pregled nedostataka i izazova hibridnih algoritama koje je svrstao u tri grupe: pravila o imenovanju, složenost hibridnih algoritama i brzina izračunavanja. Obzirom da hibridizacija podrazumeva kombinovanje dva ili više algoritama, ne postoji konzistentnost u imenovanju takvih algoritama među istraživačima. Tako na primer, akronim za jedan hibrid koji, kombinuje metaheuristike ACO i GA, je GAAPI (hybrid ant colony-genetic algorithm) [94]. Ovaj naziv unosi konfuziju među istraživače. Takođe, naziv hibrida PSO i BFO algoritama ima krajnje konfuznu skraćenicu na engleskom jeziku HPSO-BFGS, koja je teška za čitanje [127].

Hibridizacija se obično izvodi tako što se u jedan algoritam ugrađuju dodatne komponente, što povećava ukupnu kompleksnost algoritma. Upravo zbog ovoga, neki istraživači iznose arugmente koji su protiv hibridizacije. U prilog ovoj konstataciji govori činjenica da se dva rada koja su među najcitiranijim bave Taguchi genetskim algoritmom [128] i hibridnim genetskim algoritmom [129], sa 305 i 294 citata respektivno (izvor is Scopus baze, poslednja provera januar 2015.). Oba algoritma pripadaju grupi integrativnih hibrida sa jednostavnijom arhitekturom.

U mnogim radovima prikazano je da hibridni algoritmi poboljšavaju optimizacione rezultate u kontekstu brzine konvergencije i kvaliteta rezultata [93]. Međutim, priloženi grafikoni konvergencije su prikazani u odnosu na broj iteracija. Zbog toga brža konvergencija ne podrazumeva eksplicitno i realno stopu konvergencije zbog činjenice da hibridni algoritmi obično koriste veći broj internih iteracija [93]. Tako

na primer, hibrid genetskog i PSO algoritma definiše jedan ciklus (iteraciju) kao zbir jedne iteracije genetskog i PSO algoritma, što u stvari predstavljaju dva ciklusa na krivi konvergencije. Da bi se ovo sprečilo, koristi se mera poput broja evaluacija funkcija. Osim navedenog, zbog povećane složenosti, hibridi pokazuju i određeni nivo prekomernog rada. Ovo svakako utiče na ukupne performanse algoritma. Ove prekomernosti, kao i vremenska kompleksnost bi trebale biti uključene radi objektivne komparativne analize.

Osim svega navedenog, postoje i druga pitanja hibridnih algoritama. Na primer, većina hibridnih pristupa povećava broj parametara osnovnih algoritma, što otežava fino podešavanje parametara za dati optimizacioni problem [93]. Takođe, povećana složenost otežava analizu koja bi trebala da odgovori na pitanje zašto hibrid poboljšava osnovni algoritam, kao i implementaciju samih pristupa, pa se tokom implementacije hibrida češće javljaju greške.

5 Unapređenje metaheuristika rojeva hibridizacijom

Hibridni algoritmi nastaju kombinovanjem dva ili više algoritama, tako što inkorporiraju prednosti jednih, dok istovremeno eliminišu nedostatke drugih algoritama. Metaheuristike rojeva koje se hibridizuju često daju mnogo bolje rezultate od osnovnih implementacija, ali se do takvih hibrida dolazi teoretskim uvidom i empirijski, gde su česti neuspesi koji takođe mogu biti evidentirani jer mogu ukazivati na strukturnu nekompatibilnost pojedinih elemenata određenih algoritama.

Svi hibridni algoritmi koji su implementirani nastali su na osnovu višegodišnjeg istraživanja, kako teoretskog, tako i empirijskog. Pre implementacije hibridnih algoritama, analizirane su i testirane njihove osnovne verzije, kako bi se i sa teorijskog i sa praktičnog aspekta utvrdile njihove prednosti i nedostaci. Posebna pažnja je posvećana jednačinama intenzifikacije i diversifikacije i balansu između procesa eksploatacije i eksploracije, pošto najčešće u ovim komponentama leže nedostaci algoritama rojeva. Za ove potrebe implementirane su i testirane brojne unapređene verzije osnovnih algoritama koje su dobijene promenama u jednačinama pretrage i optimizacijom pojedinih parametara (lokalnih ili globalnih). O ovome svedoči veliki broj objavljenih radova u međunarodnim časopisima i na međunarodnim konferencijama.

Nakon utvrđivanja prednosti i nedostataka svakog algoritma roja, pristupilo se njihovoj hibridizaciji. Da bi se pronašao optimalan skup vrednosti kontrolnih parametara hibridne metaheuristike, vršen je veliki broj testiranja, po principu „pokušaja i greške".

Implementirane su i prikazane hibridizacije veštačke kolonije pčela (eng. artificial bee colony - ABC), optimizacije pretragom svitaca (eng. firefly algorithm - FA), optimizacije pretragom tragača (eng. seeker optimization algorithm - SOA), algoritma vatrometa (eng. fireworks algorithm - FWA) i algoritma slepog miša (bat algorithm - BA) za rešavanje problema globalne optimizacije. Predložene hibridizacije su testirane na način uobičajen u literaturi radi uporedivosti sa ostalim vodećim algoritmima: na standardnim skupovima brojnih i raznovrsnih test funkcija, bez

podešavanja parametara za pojedinačne funkcije i sa istim brojem izračunavanja funkcije cilja, ali i primenama na pojedine važne i u literaturi zastupljene praktične probleme. Standardne benčmark funkcije koje se koriste testirane su i sa malim, i sa velikim brojem varijabli (problemi velikog obima), tako da se u upoređivanju sa poznatim rezultatima iz literature koristi i taj elemenat.

5.1 Hibridizacija metaheuristike ABC

5.1.1 Osnovna metaheuristika ABC

ABC je najpoznatiji algoritam iz grupe metaheuristika koje simuliraju rojeve pčela. Ovaj pristup je prvi predložio Karaboga [51] za probleme neprekidne optimizacije bez ograničenja, a kasnije su ga razvili Karaboga i Bastuk [52], [53].

U ABC algoritmu proces pretrage vrše tri vrste veštačkih pčela (agenata), i to: pčele radilice, pčele posmatrači i izviđači (skauti). Pčele posmatrači čekaju u košnici u okviru „podijuma za igranje", dok pčele radilice eksploatišu izvor hrane (cvet) koji su već ranije posetili. Izviđači izvode proces slučajne pretrage. Polovina populacije ABC algoritma sastoji se iz pčela radilica, a druga polovina iz pčela posmatrača. Svakom cvetu (potencijalnom rešenju problema) pridružuje se samo jedna pčela radilica. Drugačije rečeno, broj pčela radilica jednak je broju potencijalnih rešenja oko košnice. Pčela radilica čiji izvor hrane presuši postaje izviđač. Dakle, pčele radilice i posmatrači izvode proces intenzifikacije, dok izviđači vrše diversifikaciju.

Glavni koraci metaheuristike ABC prikazani su u Algoritmu 9 [53]:

Algoritam 9 Pseudo-kod osnovnog algoritma ABC

```
inicijalizuj početnu populaciju
evaluacija populacije
ciklus = 1
repeat
    faza pčele radilice
    izračunavanje verovatnoće za pčele posmatrače
    faza pčela posmatrača
    faza pčela izviđača
    memorisanje najboljeg rešenja
    ciklus = ciklus + 1
until nije dostignut najveći broj ciklusa
```

Kao što se vidi iz prikazanog pseudo-koda, svaki ciklus pretrage sastoji se iz tri koraka [53]: slanje pčela radilica do izvora hrane i merenje njegovog kvaliteta, selekcija izvora hrane za eksploataciju od strane pčele posmatrača na osnovu dobijenih informacija od pčela radilica i merenje količine nektara na tim izvorima hrane (kvalitet rešenja) i određivanje pčela izviđača i slanje na slučajne izvore hrane. U fazi inicijalizacije algoritma, pozicije izvora hrane se slučajno biraju i meri se količina nektara na njima. Količina nektara se modelira pomoću funkcije podobnosti koja zavisi od vrednosti funkcije cilja. Nakon toga, pčele radilice koje su u inicijalizaciji pridružene za određena rešenja vraćaju se u košnicu i dele informacije o njihovom kvalitetu sa pčelama posmatračima koje čekaju na u košnici. U sledećoj fazi, nakon što podele informacije sa pčelama posmatračima, pčele radilice se vraćaju do cveta koji su ranije posetili i biraju novi cvet za eksploataciju u njegovom susedstvu na osnovu vizuelnih informacija. U trećoj fazi, pčele posmatrači biraju cvet za eksploataciju na osnovu količine nektara. Sa povećanjem količine nektara na cvetu, raste i verovatnoća da će posmatrači izabrati određeni cvet. Zatim, pčele posmatrači na isti način kao i pčele radilice biraju cvet u susedstvu izabranog cveta na osnovu vizuelnih informacija. Kada određeni cvet presuši, taj cvet biva napušten, a pčela koja ga je eksploatisala postaje izviđač koji bira slučajni cvet. U originalnom ABC pristupu [53], u nekim ciklusima makar jedna pčela postaje izviđač.

U ABC algoritmu, pozicija izvora hrane (cvet) predstavlja potencijalno rešenje op-

timizacionog problema, dok količina nektara na njemu modelira njegov kvalitet (podobnost). U prvom koraku, generiše se slučajno distribuirana populacija od SN rešenja, gde SN označava veličinu populacije. Svako potencijalno rešenje (izvor hrane) je D-dimenzioni vektor x_i $(i = 1, 2, ...SN)$, gde je D broj parametara problema koji se optimizuje. Nakon inicijalizacije, ABC se ponavalja u više ciklusa $C = 1, 2, ..., C_{max}$ procesa pretrage pčela radilica, posmatrača i izviđača. Pčele radilice i posmatrači sa određenom verovatnoćom modifikuju postojeće rešenje (poziciju) i pronalaze novo rešenje u susedstvu postojećeg i mere njegovu podobnost. Ako je podobnost novog rešenja bolja od starog, staro rešenje se odbacuje i zamenjuje se novim. U suprotnom, pčela zadržava staro rešenje i nastavlja da ga eksploatiše. Nakon što sve pčele radilice završe svoj proces pretrage, one razmenjuju informacije o podobnosti sa pčelama posmatračima, koje na osnovu dobijenih informacija na isti način kao i pčele radilice vrše svoj proces pretrage.

Posmatrači biraju rešenje za eksplotaciju i sa verovatnoćom p_i na osnovu njegove podobnosti fit_i prema sledećem izrazu [53]:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}, \tag{5.1}$$

gde je fit_i vrednost funkcije podobnosti rešenja i koju je izračunala pčela radilica i koja je proporcionalna količini nektara cveta na poziciji i, a SN je broj izvora hrane koji je jednak broju pčela radilica (BN) u populaciji.

Prilikom generisanja novog rešenja u susedstvu starog, koristi se sledeći izraz [53]:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{5.2}$$

gde su $k \in 1, 2, ..., SN$ i $j \in 1, 2, ...D$ slučajno izabrani indeksi. Bez obzira što je indeks k izabran slučajno, on mora da zadovoljava uslov da je različit od i. Parametar ϕ_{ij} je pseudo-slučajan broj iz intervala $[-1, 1]$ i on kontroliše proces generisanja novog rešenja u susedstvu rešenja x_{ij} na osnovu vizuelnih informacija pčela.

Iz jednačine (5.2), vidi se da se sa smanjivanjem razlike između parametara x_{ij} i x_{kj} smanjuje i petrubacija pozicije x_{ij}. Dakle, kako proces pretrage konvergira ka

optimalnom domenu, dužina koraka se smanjuje [53]. Ako nova vrednost parametara generisanih na ovaj način ispadne iz predefinisanih granica, vrednost se vraća u dopustivi opseg.

Kao što je već rečeno, kada određeni izvor hrane presuši, on se zamenjuje novim slučajnim izvorom koji generiše pčela izviđač. Broj iteracija algoritma nakon koga izvor hrane ne može da se poboljša i nakon koga se zamenjuje naziva se *limit* i on predstavlja kontrolni parametar algoritma.

ABC algoritam koristi četiri procesa selekcije [53]:

- globalnu selekciju koju izvode pčele posmatrači radi pronalaženja delova prostora pretrage koji "obećavaju" prema jednačini (5.1);
- lokalnu selekciju koju vrše pčele radilice i posmatrači na osnovu lokalnih informacija;
- lokalnu selekciju koja se naziva gramziva selekcija i koju izvode sve pčele, a koja se implementira tako što se zadržava ono rešenje koje ima veću vrednost funkcije podobnosti i
- pseudo-slučajni proces selekcije koji izvode pčele izviđači.

U ABC metaheuristici postoje tri osnovna kontrolna parametra i to su: broj izvora hrane koji jednak broju pčela radilica i posmatrača (SN), parametar *limit* i maksimalni broj ciklusa algoritma (MCN).

ABC je jedna od najšire korišćenih metaheuristika inteligencije rojeva i u literaturi može da se nađe veliki broj implementacija [78].

5.1.2 Nedostaci metaheuristike ABC

Ispitivanjem i praktičnim eksperimentima uočeno je da proces eksploatacije ABC algoritma nije dovoljno intenzivan i da algoritam konvergira previše sporo ka optimalnom regionu prostora pretrage. Nakon velikog broja ciklusa izvršavanja algoritma, kada je optimum skoro pronađen, ovaj nedostatak je još izraženiji zbog slučajne pretrage pčele izviđača.

Kao i kod mnogih drugih metaheuristika rojeva, osnovna jednačina pretrage ABC

algoritma (jednačina 5.2) bazira se na operatoru mutacije, bez eksplicitne primene operatora ukrštanja, a proces selekcije se vrši na osnovu podobnosti individua. Iz jednačine (5.2) vidi se da se novo rešenje generiše prostom mutacijom varijable trenutnog rešenja. Vrednost varijable x_{ij} se menja u iznosu $\phi_{ij}(x_{ij} - x_{kj})$, gde je x_{kj} vrednost varijable rešenja koje se nalazi u okolini trenutnog rešenja x_i. Dakle, intenzifikacija koja se izvodi bez primene operatora ukrštanja utiče na sporu konvergenciju ABC algoritma.

Sporu konvergenciju ABC algoritma pojačava i to što je uočeno da procesi intenzifikacije i diversifikacije nisu dovoljno razdvojeni. U nekim slučajevima jednačina intenzifikacije (5.2) vrši pretragu koja je po svojim karakteristikama bliža globalnoj (diversifikacija), nego lokalnoj (intenzifikacija). Tako na primer, ako je vrednost parametra ϕ_{ij} bliža vrednosti -1 ili 1 i ako je razlika $x_{ij} - x_{kj}$ velika, nejasno je da li ova jednačina vrši intenzifikaciju ili diversifikaciju. U početnim fazama izvršavanja algoritma razlika $x_{ij} - x_{kj}$ je veća pošto je varijetet jedinki u populaciji veliki. Zatim se ova razlika smanjuje, kako algoritam konvergira ka optimalnom domenu i kako jedinke postaju sličnije.

U prilog tvrdnji da jednačina (5.2) često izvodi proces diversifikacije ide i to da se praktičnim eksperimentima utvrdilo da se prilikom generisanja novog rešenja u okolini postojećeg (i u slučaju optimizacije sa i bez ograničenja) često dešava da varijable ispadaju iz opsega dozvoljenih granica. U takvim situacijama je potrebno da se primeni mehanizam vraćanja vrednosti varijabli u dozvoljeni opseg, što nepotrebno produžava vreme izvršavanja algoritma.

Obzirom da se sa smanjivanjem diversifikacije rešenja u populaciji u kasnijim iteracijama izvršavanja algoritma efekat eksploracije jednačine (5.2) smanjuje, logičan zaključak je da se nedostatak spore konvergencije ne ispoljava u kasnijim iteracijama. Međutim, u kasnijim ciklusima izvršavanja, iako je pronađen optimalni domen prostora pretrage, u osnovnom ABC algoritmu veliki broj pčela izviđača i dalje izvodi eksploraciju, koja je u ovom slučaju nepotrebna. Zbog ovoga je problem spore konvergencije izražen tokom celog izvršavanja ABC algoritma. Ispitivanjem je utvrđeno da je problem spore konvergencije ozbiljniji u kasnijim fazama, pošto bi tada algo-

ritam trebao da vrši finu pretragu u optimalnom domenu.

Osim spore konvergencije, navedeni problemi uzrokuju i problem neusklađenog balansa između intenzifikacije i diversifikacije, koji je naglašeniji u ranijim iteracijama, pošto u nekim ciklusima jednačina intenzifikacije više "vuče" na stranu diversifikacije. Ovaj problem je s druge strane ozbiljniji u kasnijim fazama, kao što je već i navedeno.

Dakle, kao osnovni nedostaci metaheuristike ABC navode se dva međusobno povezana problema - spora konvergencija ka optimalnom domenu prostora pretrage i neodgovarajući balans između intenzifikacije (eksploatacije) i diversifikacije (eksploracije). Prvi problem se javlja zbog odsustva operatora ukrštanja i nedefinisanosti kada i u kojoj meri jednačina intenzifikacije izvodi ovaj proces.

S druge strane, metaheuristika ABC je efikasna u izvođenu procesa diversifikacije i zato se retko dešava da se algoritam zaglavi u suboptimalnim regionima. Upravo zbog ove prednosti, osnovni ABC postiže dobre rezultate u velikom broju problema globalne optimizacije sa i bez ograničenja.

5.1.3 Hibridizacija metaheuristike ABC i genetskih operatora

Metaheuristika ABC generiše dobra rešenja za široku lepezu benčmark [52], [53] i praktičnih problema [78]. Međutim, osnovna verzija algoritma pokazuje dva nedostatka: sporu konvergenciju i neusklađeni balans između eksploatacije i eksploracije.

Da bi se ispravili navedeni nedostaci, pošlo se od predpostavke da se slabosti osnovnog algoritma ABC mogu otkloniti hibridizacijom. Predloženi hibrid metaheuristike ABC i genetskih operatora (eng. genetically inspired artificial bee colony - GI-ABC), čiji je opis prikazan u objavljenom radu, rešava problem neuravnoteženog balansa između intenzifikacije i diversifikacije u kasnijim iteracijama izvršavanja algoritma [130]. GI-ABC je predložen za probleme sa ograničenjima.

Hibridna metaheuristika GI-ABC uvodi trostruke modifikacije u osnovni ABC algoritam za optimizaciju bez ograničenja [53]: inkorporiranje dodatnog kontrolnog parametra u metodu intenzifikacije po ugledu na jedan predložen pristup [66], prila-

gođavanje algoritma za rešavanje globalne optimizacije sa ograničenjima i promena procesa pretrage u kasnijim fazama izvršavanja uvođenjem genetskih operatora.

U fazi inicijalizacije svakoj pčeli radilici iz populacije dodeljuje se slučajno generisani izvor hrane (potencijalno rešenje optimizacionog problema). Svako rešenje x_i $(i = 1, 2, ...SN)$ je D-dimenzioni vektor, gde SN označava veličinu populacije, a D označava broj varijabli problema. Početna populacija potencijalnih rešenja se kreira pomoću sledećeg izraza:

$$x_{i,j} = lb_j + rand(0, 1) * (ub_j - lb_j), \tag{5.3}$$

gde je $x_{i,j}$ j-ta varijabla i-tog rešenja u populaciji, $rand(0, 1)$ je pseudo-slučajan broj između 0 and 1, a ub_j i lb_j su gornja i donja granica j-te varijable, respektivno.

Jednačina pretrage ABC algoritma (5.2) modifikovana je uvođenjem dodatnog kontrolnog parametra. Novi parametar naziva se stopa modifikacije (eng. modification rate - MR) i koristi se za određivanje intenziteta procesa eksploatacije. Uz navedenu modifikaciju, jednačina pretrage GI-ABC algoritma data je sledećim izrazom:

$$v_{i,j} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \text{ ako je } R_j < MR \\ x_{ij}, \text{ u suprotnom,} \end{cases} \tag{5.4}$$

gde je $k \in 1, 2, ..., SN$ slučajno izabrani indeks, ϕ_{ij} je pseudo-slučajan broj u intervalu $[-1, 1]$, a R_j je uniformno distribuirani realan broj iz opsega $[0, 1]$. Parametru MR dodeljuje se vrednost iz intervala $[0, 1]$.

Kao druga razlika u odnosu na osnovni ABC navodi se da je predloženi hibrid adaptiran za rešavanje problema globalne optimizacije sa ograničenjima [130]. Da bi se algoritam uspešno adaptirao za ovu vrstu problema, proces gramzive selekcije između starog i novog rešenja zamenjuje se sa Debovim pravilima [10] i na taj način se proces pretrage usmerava ka dopustivom regionu.

Debov metod koristi operator turnirske selekcije, gde se dva rešenja upoređuju u određenom vremenskom trenutku primenom sledećih kriterijuma [10]:

- svakog dopustivo rešenje i $(CV_i \leq 0)$ preferira se u odnosu na bilo koje nedo-

pustivo rešenje j ($CV_j > 0$), pri čemu je rešenje i dominantno;

- između dva dopustiva rešenja i i j ($CV_i \leq 0$ i $violation_j \leq 0$) bira se ono rešenje koje ima bolju vrednost funkcije cilja, a u slučaju minimizacije, to je $f_i < f_j$, pri čemu je rešenje i dominantno i

- između dva nedopustiva rešenja i i j ($CV_i > 0$ i $CV_j > 0$), bira se ono koje manje narušava ograničenja prostora pretrage, a to je rešenje i ako je ispunjen uslov da je $CV_i < CV_j$, pri čemu je rešenje i dominantno.

Zbog optimizacije sa ograničenjima, menja se i način na koji se računa verovatnoća pčela posmatrača. GI-ABC koristi jednačinu verovatnoće koja je predložena u jednom objavljenom radu [66]:

$$p_i = \begin{cases} 0.5 + (\frac{fitness_i}{\sum_{i=1}^{SN} fitness_i}), \text{ ako je rešenje dopustivo} \\ (1 - \frac{CV}{\sum_{i=1}^{SN} CV}) * 0.5, \text{ ako rešenje nije dopustivo,} \end{cases} \tag{5.5}$$

gde se CV računa kao:

$$CV_i = \sum_{g_j(x_i)>0} g_j(x_i) + \sum_{j=q+1}^{m} h_j(x_i), \tag{5.6}$$

CV_i je vrednost kojom rešenje x_i narušava ograničenja, $g_j(x_i)$ su ograničenja nejednakosti, a $h_j(x_i)$ su ograničenja jednakosti za rešenje x_i, $fitness_i$ je vrednost funkcije podobnosti rešenja x_i koja je direktno proporcionalna količini nektara na izvoru hrane i računa se prema sledećoj jednačini za probleme minimizacije [66]:

$$fitness_i = \begin{cases} \frac{1}{1+f_i}, \text{ ako je } f_i > 0 \\ 1 + |f_i|, \text{ u suprotnom,} \end{cases} \tag{5.7}$$

gde je f_i vrednost funkcije cilja koja se optimizuje.

Konačno, treća i najvažnija razlika između originalnog [53] i predloženog [130] pristupa odnosi se na mehanizam zamene istrošenih izvora hrane (rešenja) u populaciji.

Nakon određenog broja ciklusa, pod predpostavkom da je algoritam pronašao domen prostora u kome se nalazi optimum, više nema potrebe za izvođenjem slučajne pretrage od strane pčele izviđača. Zbog toga se proces slučajne pretrage zamenjuje, sa određenom verovatnoćom, usmerenom pretragom u okolini najboljih rešenja u populaciji. Vremenska tačka u izvršavanju algoritma nakon koje se meahnizam pčela izviđača zamenjuje sa novim mehanizmom vođenih posmatrača (eng. guided onlookers), koji izvode jaku intenzifikaciju oko najboljeg rešenja u populaciji, dobila je naziv tačka preloma (eng. breakpoint - *bp*).

Novi proces zamene istrošenih rešenja u kasnijim fazama izvršavanja algoritma koristi operatore ukrštanja i mutacije koji su „pozajmljeni" iz GA [130]. Hibridizacija sa operatorom ukrštanja predložena je iz razloga što ABC u svojoj jednačini pretrage eksplicitno ne primenjuje ovaj operator, a njegovim korišćenjem može da se poboljša brzina konvergencije.

GI-ABC usvaja filozofiju GA tako što posmatra potencijalno rešenje, na primer funkciju koja se optimizuje, kao hromozom koji se sastoji iz gena. Svaki gen modelira jednu varijablu funkcije. U ovoj implementaciji usvojen je operator uniformnog ukrštanja, gde se za svaku poziciju gena, geni između dva roditelja zamenjuju sa nezavisnom verovatnoćom p. Za razliku od ukrštanje preko jedne i više tačaka prekida, kod uniformnog ukrštanja roditelji učestvuju u kreiranju potomka na nivou gena, a ne na nivou čitavog segmenta. Glavni uticaj na formiranje potomaka ima vrednost verovatnoće p. Ako je ova vrednost oko 0.5, onda se geni razmenjuju češće između dva roditelja. U ovom slučaju, moć diversifikacije uniformnog ukrštanja je veća i proces pretrage pokazuje globalne karakteristike. S druge strane, ako je vrednost p bliže 0 ili 1, onda se manji broj gena razmenjuje između roditelja i proces pretrage pokazuje lokalne karakteristike.

Verovatnoća sa kojom se nakon *bp* broja iteracija mehanizam pčela izviđača zamenjuje sa usmerenim posmatračima inkorporirana je u GI-ABC kao dodatni kontrolni parametar koji se naziva stopa zamene (eng. replacement rate - *rr*) [130]. Potrebno je pažljivo odrediti vrednost ovog parametra, pošto on utiče na balans između intenzifikacije i diversifikacije. Ovaj parametar ne može da kontroliše korisnik.

Ako se zamena dogodi, onda se u procesu ukrštanja koriste najbolja i slučajno odabrana individua iz populacije. Utvrđeno je da algoritam postiže najbolje rezultate ako se ukrštanje izvodi češćom zamenom gena između roditelja. U GI-ABC implementaciji procesom ukrštanja generiše se samo jedan potomak [130]. U suprotnom slučaju, ako se rr uslov ne zadovolji, rešenje koje je istrošeno zamenjuje se sa slučajnim rešenjem pomoću jednačine (5.3), tj. pomoću klasičnog mehanizma izviđača. Na ovaj način se smanjuje verovatnoća da će algoritam da se zaglavi u nekom od lokalnih optimuma, zato što se u nekim slučajevima proces diversifikacije ipak izvšava.

Nakon ukrštanja, generisani potomak se izlaže procesu mutacije. Svaki gen (varijabla funkcije) se mutira sa malom verovatnoćom prema jednačini (5.8). Stopa verovatnoće mutacije (eng. mutation probability rate - mpr) je još jedan kontrolni parametar algoritma koji ne može da se podešava.

$$offsp[i] = offsp[i] + \phi 1 * (rndsol[i] - offsp[i]), \tag{5.8}$$

gde je $offsp[i]$ i-ti gen potomka, a mutacija se primenjuje sa mpr verovatnoćom na sve gene, $rndsol[i]$ je i-ti gen slučajnog rešenja iz populacije i $\phi 1$ je pseudo-slučajan broj između -0.1 i 0.1.

Kao što se vidi iz jednačine (5.8), mutacija blago skreće usmerenog posmatrača sa najboljeg rešenja ka slučajnom rešenju iz populacije [130]. Empirijskim putem je utvrđeno da je mutacija potrebna zato što se ponekada dešava da određena varijabla ima fiksnu vrednost, što onemogućava finu pretragu globalnog optimuma.

Kada algoritam uđe u najkasnije faze izvršavanja (veliki broj ciklusa), uvodi se još jedna prelomna tačka, koja se naziva druga prelomna tačka (eng. second break point - sbp). Razlika u izvršavanju algoritma nakon bp i nakon sbp broja ciklusa je u načinu na koji se kreira rešenje potomak. Nakon sbp ciklusa, usmereni posmatrač kombinuje dva najbolja rešenja u populaciji [130]. Uz predpostavku da je algoritam skoro pronašao optimalno rešenje, ovaj mehanizam izvodi još jaču intenzifikaciju. Nakon operatora ukrštanja, primenjuje se isti operator mutacije kao i u prethodnom slučaju.

U predloženom hibridu, utvrđeno je da se optimalna vrednost za *sbp* izračunava kao $sbp = bp * 1.7$.

Osim svih modifikacija originalnog ABC algoritma, potrebno je da se spomene i da GI-ABC koristi softisticirani mehanizam za upravljanje ograničenjima jednakosti. Većina optimizacionih algoritama proces pretrage počinje sa slučajnim, nedopustivim rešenjima, koja ispadaju iz dopustivog domena procesa pretrage, sa očekivanjem da će tokom izvršavanja algoritma rešenja da dostignu dopustiv domen. Međutim, ograničenja jednakosti predstavljaju težak zadatak za optimizacioni metod pošto ona smanjuju dopustiv prostor, tako da je on veoma mali u odnosu na ceo prostor pretrage. Zbog toga se ograničenja jednakosti zamenjuju sa ograničenjima nejednakosti sa malom granicom prekoračenja $\varepsilon > 0$ [131]:

$$|h(x)| - \varepsilon \leq 0, \tag{5.9}$$

Kvalitet rezultata zavisi od izbora vrednosti za granicu prekoračenja. Ako je ova vrednost previše mala, algoritam bi mogao da "promaši" dopustiva rešenja, a ako je ova vrednost velika, rezultati bi mogli da budu daleko od dopustivog domena.

Pristup koji obećava u upravljanju ograničenjima jednakosti podrazumeva dinamičko, samoadaptivno podešavanje tolerancije [132]. Kada algoritam počne da se izvršava, vrednost tolerancije ε je velika, a nakon toga se postepeno, sa svakim ciklusom, smanjuje. Dakle, u početku se veliki broj rešenja generiše van dopustivog prostora, što olakšava algoritmu da pronađe region koji obećava. Kada je algoritam dovoljno istražio prostor, a možda i pronašao optimalni region, velika odstupanja od dopustivog prostora se ne dozvoljavaju.

Dinamička podešavanja vrednosti prekoračenja ε se definišu sledećom jednačinom [132]:

$$\varepsilon(t+1) = \frac{\varepsilon(t)}{dec}, \tag{5.10}$$

gde je t brojač tekuće iteracije, a $dec > 1$ je stopa po kojoj se ε smanjuje u svakom ciklusu. U GI-ABC, donja granica za ε je postavljena na 0.0001.

Na osnovu svega navedenog, GI-ABC može da se opiše pseudo-kodom, koji je dat u Algoritmu 10 [130].

Algoritam 10 GI-ABC pseudo-kod

inicijalizuj početnu populaciju rešenja pomoću (5.3)
evauliraj populaciju
ako postoji ograničenje jednakosti, onda postavi da je $\varepsilon = 1$
$ciklus$=1
while $ciklus = MCN$ **do**
  kreiraj novo rešenje za pčelu radilicu pomoću jendačine (5.4) i evauliraj ga
  primeni selekciju između starog i novog rešenja pomoću Debovih pravila [10]
  izračunaj verovatnoće p_i pčela posmatrača pomoću (5.5) i (5.6)
  za svakog posmatrača, kreiraj novo rešenje v_i pomoću (5.4) i evauliraj ga
  primeni selekciju između starog i novog rešenja pomoću Debovih pravila [10]
  odredi napuštena rešenja pomoću parametra $limit$ i ako postoje zameni ih sa:
    a) slučajnim rešenjem pomoću jednačine (5.3), ako je $ciklus < bp$
    b) rešenjem potomkom najboljeg i slučajnog rešenja ako je $bp \leq ciklus \leq sbp$
      i ako je rr uslov zadovoljen, a u suprotnom slučaju isto kao i pod a)
    c) rešenjem potomkom dva najbolja rešenja u populaciji, ako je $sbp < ciklus$
      i ako je rr uslov zadovoljen, a u suprotnom slučaju isto kao i pod a)
  memoriši najbolje pronađeno rešenje do sada
  ako je $\varepsilon > 0.0001$, onda smanji ε prema (5.10)
  izračunaj ograničenja potencijalnih rešenja korišćenjem nove ε vrednosti
  $ciklus = ciklus + 1$
end while

5.1.3.1 Eksperimentalni rezultati

Da bi se utvrdila efikasnost metaheuristike GI-ABC izvršena su dva skupa eksperimenata. Prvo su izvršena testiranja sa različitim podešavanjima kontrolnih parametara algoritma, kako bi se utvrdilo sa kojim vrednostima parametra se uspostavlja najbolji balans između intenzifikacije i diversifikacije. U okviru drugog seta eksperimenata prikazana je komparativna analiza sa nekim od najboljih algoritama koji se mogu naći u svetskoj literaturi. Svi testovi vršeni su na dobro poznatim G benčmark problemima za globalnu optimizaciju sa ograničenjima, koji su prvo predloženi za potrebe jednog svetskog kongresa [12].

Platforma koja se koristila za eksperimente je Intel Core 2 Duo T8500 procesor sa 4GHz i 4GB RAM memorije, Windows 7 x64 Ultimate operativnim sistemom i Visual Studio 2010 .NET Framework 4.0 okruženjem. Ponekada se u svetskoj literaturi navodi platforma na kojoj su testirani algoritmi, kako bi mogla da se izvrši komparacija brzine izvršavanja različitih algoritama. Takođe, s obzirom da različite platforme koriste različite generatore pseudo-slučajnih brojeva, navođenje platforme omogućava da se reprodukuju isti rezultati na osnovu istog semena pseudo-slučajnog broja.

Isti vrednosti osnovnih kontrolnih parametara kao i u ABC pristupu za optimizaciju sa ograničenjima (Karaboga i Akay, 2011) [66] korišćene su, kako bi se kvantitativno izrazila unapređenja hibridnog algoritma u odnosu na osnovni ABC pristup za optimizaciju sa ograničenjima.

Vrednost stope modifikacije MR je postavljena na 0.8, veličina kolonije SN na 40, a najveći broj ciklusa u jednom izvršavanju algoritma MCN na 6,000 (u radu se koristi američki način prikazivanja brojeva sa zarezom, kao što je praksa u svetskoj literaturi). Dakle, korišćeno je 240,000 evaluacija funkcija. Vrednost parametra $limit$ je postavljena na 150 (MCN/SN). Parametri za upravljanje ograničenjima jednakosti su $\varepsilon = 1$ i $dec = 1.002$.

GI-ABC kontroliše balans između intenzifikacije i diversifikacije podešavanjem vrednosti bp i rr parametara. Promenom ovih vrednosti menjaju se i najbolje i prosečne vrednosti koje algoritam generiše u jednom izvršavanju. GI-ABC koristi fiksne vrednosti ovih parametara koje su utvrđene empirijskim putem. Parametar mpr povremeno ima pozitivan uticaj i na najbolje i na prosečne vrednosti tako što pomaže algoritmu da se "izbavi" iz lokalnog optimuma.

Sa smanjivanjem vrednosti bp i povećanjem rr, intenzifikacija se povećava, a diversifikacija se smanjuje, i obrnuto. Na osnovu izvršenih testova, u najvećem broju slučajeva su se sa smanjenjem bp i povećavanjem rr najbolja otkrivena rešenja u toku 30 puštanja algoritma poboljšala, dok su se prosečna rešenja pogoršala. Ovo je posledica toga da ako alogoritam uspe u ranim ciklusima da pronađe pravi deo prostora pretrage (sa malo snage eksploracije), dobra rešenja se kombinuju i bolja

rešenja se generišu. Međutim, ako algoritam promaši optimalni domen prostora pretrage u ranim ciklusima, kombinovanjem loših rešenja dobijaju se lošiji rezultati. Tako na primer, ako u 5 od 30 puštanja algoritma, algoritam otkrije pravi deo prostora, dobra najbolja rešenja se postižu u 5 puštanja, ali u ostalih 25, najbolja rešenja su daleko od pravog optimuma, što u proseku vodi ka lošijim prosečnim rezltatima.

S druge strane, povećavanjem bp i smanjivanjem rr, najbolji rezultati se pogoršavaju, dok se prosečni poboljšavaju. U ovom slučaju, algoritam ima više ciklusa u kojima može da pronađe pravi deo prostora pretrage, ali ima manji broj ciklusa za kombinovanje dobrih rešenja ako otkrije optimalni deo prostora pretrage u ranim ciklusima izvršavanja.

Za potrebe analize konvergencije GI-ABC algoritma, izvršen je čitav niz eksperimenata sa različitim vrednostima parametara bp, rr, mpr, sbp i p, kako bi se utvrdio najbolji balans intenzifikacije i diversifikacije i njegovo uticaj na najbolje i prosečne vrednosti. Parametar sbp zavisi od bp, dok p ima fiksiranu vrednost. U Tabeli 5.1 prikazani su rezultati sa različitim vrednostima bp parametra. Kao indikatori performansi korišćeni su najbolji i prosečni rezultati. Ostali parametri su fiksirani na sledeće vrednosti: rr na 0.9, mpr na 0.01, a sbp na $1.7 * bp$.

Svi eksperimenti su ponavljani 30 puta i u svakom izvršavanju algoritma korišćeno je različito seme pseudo-slučajnog broja. Kao što je i očekivano, svaka funkcija je specifična i ponaša se različito, ali bez obzira na to, opšte pravilo može da se izvede. Sa povećanjem bp vrednosti, snaga intenzifikacije se smanjuje, a diversifikacije raste. Implikacija ovoga je da bi najbolji rezultati trebalo da se pogoršavaju, a prosečni da se poboljšavaju.

Iz Tabele 5.1, vidi se da promena bp parametra ima najveći uticaj na $G2$ problem. Najbolji rezultat je isti za $bp = 1,000$ i $bp = 2,000$ ali se pogoršava kada je $bp = 3,000$. Prosečne vrednosti se značajno poboljšavaju od $bp = 1,000$ do $bp = 3,000$. U testu sa $bp = 500$, i prosečni i najbolji rezultati znatno lošiji nego u svim ostalim slučajevima, zato što algoritam u 500 ciklusa nije uspeo da konvergira ka optimalnom delu prostora pretrage.

Tabela 5.1: Rezultati sa različitim vrednostima za *bp* parametar [130]

Problem	Optimum		bp=500	bp=1000	bp=2000	bp=3000
G1	-15.000	Najbolji	-15.000	-15.000	-15.000	-15.000
		Prosečni	-15.000	-15.000	-15.000	-15.000
G2	-0.803619	Najbolji	-0.7946601	-0.803618	**-0.803618**	-0.803614
		Prosečni	-0.7202719	-0.765693	-0.798054	**-0.800151**
G3	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000
		Prosečni	-1.000	-1.000	-1.000	-1.000
G4	-30665.539	Najbolji	-30665.539	-30665.539	-30665.539	-30665.539
		Prosečni	-30665.539	-30665.539	-30665.539	-30665.539
G5	5126.497	Najbolji	5126.497	5126.497	5126.497	5126.497
		Prosečni	5258.619	5221.603	5209.028	**5164.762**
G6	-6961.814	Najbolji	-6961.814	-6961.814	-6961.814	-6961.814
		Prosečni	-6961.814	-6961.814	-6961.814	-6961.814
G7	24.306	Najbolji	24.306	24.306	24.306	24.306
		Prosečni	24.513	24.481	24.421	**24.395**
G8	-0.095825	Najbolji	-0.095825	-0.095825	-0.095825	-0.095825
		Prosečni	-0.095635	-0.095825	-0.095825	-0.095825
G9	680.63	Najbolji	680.631	680.631	680.630	680.630
		Prosečni	680.648	680.646	680.641	**680.635**
G10	7049.25	Najbolji	7049.282	7049.282	**7049.282**	7049.282
		Prosečni	7383.949	7282.871	7250.293	**7192.397**
G11	0.75	Najbolji	0.750	0.750	0.750	0.750
		Prosečni	0.750	0.750	0.750	0.750
G12	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000
		Prosečni	-1.000	-1.000	-1.000	-1.000
G13	0.053950	Najbolji	0.055	0.054	0.054	0.054
		Prosečni	0.335	0.305	0.279	**0.248**

U optimizaciji funkcije $G9$, optimum je postignut tek kada je vrednost *bp* postavljena na 2,000. Dakle, u ovom slučaju, algoritam nije uspeo da konvergira ka optimalnom domenu za manji broj ciklusa. U svim eksperimentima za $G10$ benčmark, algoritam postiže najbolje rezultate koji su jako blizu optimumu. Nažalost, sa vrednostima *bp* koje su veće od 3,000 najbolji rezultati se pogoršavaju. Kao što je i očekivano,

u ovom testu se prosečne vrednosti poboljšavaju sa $bp = 500$ do $bp = 3,000$. Zaključuje se da za problem $G10$ algoritam uspeva da pogodi pravi domen prostora u ranijim ciklusima. U testu $G13$, algoritam konvergira ka optimalnom rešenju u svim testovima osim za $bp = 500$. Zaključuje se da 500 ciklusa nije dovoljno da se pronađe prostor u kome se nalazi optimalno rešenje.

Dakle, GI-ABC postiže najbolje rezultate sa vrednostima $bp = 1,000$ i $bp = 2,000$. Međutim, najbolji balans između najboljih i prosečnih rezultata se postiže u testovima sa $bp = 3,000$, tako što se uz malo žrtvovanje najboljih rezultata postižu mnogo bolje prosečne vrednosti. U Tabeli 5.1, najbolja rešenja iz svake kategorije su označena masnim slovima, odakle se i potvrđuje prethodna konstatacija. Zbog ovog razumnog kompromisa najboljih i prosečnih rezultata, fiksirana je vrednost bp na 3,000 i sa tom vrednošću je algoritam upoređivan sa drguim metaheuristikama.

Kao što je slučaj i sa standardnim metaheuristikama, i parametre hibridnih algoritama je potrebno pažljivo podesiti, kako bi radili sa punim potencijalom. Podešavanja se vrše empirijskim putem, baš kao što je i prikazano za metaheuristiku GI-ABC.

GI-ABC je direktno upoređivan sa originalnim ABC za globalnu optimizaciju sa ograničenjima (Karaboga i Akay, 2011) [66], ABC algoritmom koji usmerava pretragu ka najboljem rešenju (eng. smart filight ABC - SF-ABC) čiji su tvorci Mezura-Montes i ostali [133], GA sa samoadaptivnom kaznenom funkcijom (eng. self-adaptive penalty function GA - SAPF-GA), koga su prikazali Tassema i Yen [134] i adaptivnim modelom evolutivne strategije (adaptive tradeoff model evolutionary strategy - ATMES) koga su prikazali Wang i ostali [135]. Komparativna analiza najboljih i prosečnih rezultata je prikazana u Tabeli 5.2. Optimalni rezultati test funkcija preuzeti su iz [66].

Na osnovu empirijskih rezultata prikazanih u Tabeli 5.2, zaključuje se da GI-ABC postiže bolje rezultate, kada se posmatraju najbolje vrednosti, od ABC algoritma za funkcije $G2$, $G7$, $G9$, $G10$ i $G13$. Najbolji rezultat za funkciju $G5$ prijavljen u [66] je bolji nego poznati optimum za taj benčmark, što je posledica narušavanja ograničenja dopustivog prostora. U testovima $G1$, $G3$, $G4$, $G6$, $G8$, $G11$ i $G12$, GI-

ABC je postigao optimalne rezultate kao i originalni ABC. U skoro svim testovima, GI-ABC postiže bolje rezultate od originalnog ABC kada su u pitanju prosečni rezultati.

Od ostalih algoritama, samo SF-ABC [133] i ATMES [135] postižu bolje prosečne vrednosti za neke funkcije. U odnosu na SF-ABC, GI-ABC postiže značajno bolje rezultate u testovima $G2$ i $G7$, dok za ostale funkcije oba algoritma postižu približno iste rezutlate. Najveća razlika performansi može da se uoči u testu $G2$, gde SF-ABC pokazuje nedostatak preuranjene konvergencije. SF-ABC postiže bolje prosečne vrednosti od GI-ABC u $G5$ i $G10$ problemima. S druge strane, GI-ABC je bolji u testovima $G1$, $G2$, $G7$, $G9$ i $G13$. Oba algoritma postižu iste vrednosti i za najbolje i za prosečne vrednosti kod relativno lakih problema, kao što su $G3$, $G4$, $G6$, $G8$ i $G12$.

GI-ABC postiže bolje vrednosti i za najbolje i za prosečne rezultate od ATMES algoritma [135] u $G2$ i $G10$ testovima. Međutim, ATMES postiže malo bolje performanse od GI-ABC u komparaciji prosečnih vrednosti kod $G5$, $G7$, $G9$ i $G13$ problemima. U testu $G7$, ATMES postiže prosečne vrednosti koje su bliske optimumu, dok u testu $G13$ pokazuje iste vrednosti i za najbolje i za prosečne rezultate. U drugim benčmark problemima, oba algoritma se ponašaju slično.

SAPF-GA se u komparativnoj analizi pokazao kao najlošiji pristup, od koga je hibrid bolji u skoro svim testovima, kako za prosečne, tako i za najbolje vrednosti.

Komparacija sa originalnim ABC je najrelevatnija, pošto ona na najbolji način prikazuje unapređenja hibridne metaheuristike. Prema navedenim rezultatima, očigledno je da hibridni algoritam poboljšava konvergenciju osnovnog (bolje prosečne vrednosti u skoro svim testovima) i da povećava kvalitet pretrage (bolji najbolji rezultati u nekim testovima). Dakle, inkorporiranjem genetskih operatora poboljšan je proces intenzifikacije u kasnijim ciklusima i uspostavljen je bolji balans između eksploatacije i eksploracije. Iz ovog primera se najbolje vidi da hibridne metaheuristike nastaju pametnom kombinacijom dobrih strukruralnih elemenata različitih pristupa, a na osnovu detaljne teorijske i prakitčne analize.

Tabela 5.2: Rezultati komparativne analize GI-ABC sa drugim algoritmima [130]

Prob.	Opt.		SF-ABC	SAPF-GA	ATMES	ABC	GI-ABC
G1	-15.000	Najbolji	-15.000	-15.000	-15.000	-15.000	-15.000
		Prosečni	-14.13	-14.552	-15.000	-15.000	-15.000
G2	-0.803619	Najbolji	-0.709034	-0.803202	-0.803388	-0.803598	**-0.803614**
		Prosečni	-0.471210	-0.755798	-0.790148	-0.792412	**-0.800151**
G3	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000	-1.000
		Prosečni	-1.000	-0.964	-1.000	-1.000	-1.000
G4	-30665.539	Najbolji	-30665.539	-30665.401	-30665.539	-30665.539	-30665.539
		Prosečni	-30665.539	-30665.922	-30665.539	-30665.539	-30665.539
G5	5126.497	Najbolji	5126.497	5126.907	5126.498	5126.484*	5126.497
		Prosečni	5126.526	5214.232	5127.648	5185.714	**5164.762**
G6	-6961.814	Najbolji	-6961.814	-6961.046	-6961.814	-6961.814	-6961.814
		Prosečni	-6961.814	-6953.061	-6961.814	-6961.814	-6961.814
G7	24.306	Najbolji	24.316	24.838	24.306	24.330	24.306
		Prosečni	24.657	27.328	24.316	24.473	**24.395**
G8	-0.095825	Najbolji	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
		Prosečni	-0.095635	-0.095825	-0.095825	-0.095825	-0.095825
G9	680.63	Najbolji	680.630	680.773	680.630	680.634	680.630
		Prosečni	680.643	681.246	**680.633**	680.640	680.635
G10	7049.25	Najbolji	7049.547	7069.981	7052.253	7053.904	**7049.285**
		Prosečni	7116.934	7238.964	7250.437	7224.407	**7192.397**
G11	0.75	Najbolji	0.750	0.750	0.750	0.750	0.750
		Prosečni	0.750	0.751	0.750	0.750	0.750
G12	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000	-1.000
		Prosečni	-1.000	-1.000	-1.000	-1.000	-1.000
G13	0.053950	Najbolji	0.054	0.054	0.054	0.760	0.054
		Prosečni	0.263	0.286	**0.054**	0.968	0.248

5.1.4 Hibridizacija metaheuristike ABC za problem planiranja RFID mreže

Hibridne metaheuristike ABC primenjene su i na praktične probleme. Iako su benč-mark problemi odlični indikatori performansi algoritma, potrebno je da se utvrdi

kako se algoritam ponaša i na pravim problemima. Praktični problemi su krajnji test i svrha implementiranja algoritma i zato su važniji nego benčmark problemi.

Dva hibridna ABC algoritma primenjena su na problem planiranja mreže identifikacije pomoću radio frekvencije (eng. radio frequency identification - RFID). Ovaj problem spada u grupu teške globalne optimizacije sa više funkcija cilja. Hibridni GI-ABC, koji je prikazan u Sekciji 5.1.3 i testiran na benčmark problemima sa ograničenjima, isproban je i na ovom važnom i relativno komplikovanom praktičnom problemu, gde se dodatno dokazuje njegov kvalitet [136]. U drugom slučaju je ABC hibridizovan sa heuristikom za određivanje broja i početne lokacije čitača u RFID mreži [137].

5.1.4.1 Formulacija RFID problema

RFID tehnologija je sve više u upotrebi poslednjih godina i široko je prihvaćena kao standard u mnogim oblastima [138]. RFID sistem se sastoji iz tagova i čitača. Čitači pristupaju informacijama koje su uskladištene na tagovima sa razdaljine od nekoliko metara.

Prilikom implementacije RFID mreže nekoliko ciljeva mora da se zadovolji, a najvažniji među njima su: optimalna pokrivenost tagova, smanjenje troškova i kvalitet servisa (eng. quality of service - QoS) [14]. Ovi izazovi su u literaturi poznati kao problem planiranja RFID mreže sa više funkcija cilja (eng. multi-objective RFID network planning problem - MRNP) [139] i kao takav predstavlja najvažniji zadatak koji treba da se reši prilikom implementacije RFID mreže sa velikim brojem tagova i čitača.

U ovom radu razmatran je problem planiranja RFID mreže sa sledećim ciljevima: maksimalna pokrivenost tagova, minimalni broj čitača, minimalna interferencija između čitača i minimalna emitovana energija. Isti ciljevi korišćeni su i u [14]. Prvi i najvažniji cilj je u suprotnosti sa ostalim. Tako na primer, da bi se ostvario cilj optimalne pokrivenosti tagova, broj čitača treba da se poveća. Sa većim brojem čitača raste emitovana energija i povećava se mogućnost za interferenciju.

U literaturi postoji mnogo pristupa za rešavanje ovog problema. Tako na primer

primenom metode težinskih koeficijenata problem planiranja RFID mreže sa više funkcija cilja rešava se kao problem sa jednom funkcijom cilja [140]. Na ovaj način se uticaj svakog cilja određuje vrednostima odgovarajućih težinskih koeficijenata. U drugim pristupima koristi se metod hijerarhijskog upravljanja ciljevima, tako što se prvo vrši optimizacija najznačajnijeg cilja, zatim sledećeg po važnosti, itd. [14].

Glavni cilj problema planiranja RFID mreže sa više funkcija cilja je da se omogući najveći stepen pokrivenosti tagova. Da bi se definisao ovaj cilj, dosmernu komunikaciju između tagova i čitača treba uzeti u obzir. Svaki tag $t \in TS$ je pokriven čitačem ako i samo ako postoji čitač $r_1 \in RS$ koji zadovoljava uslov da je $PT_{r_1,t} \geq T_t$ i čitač $r_2 \in RS$ koji zadovoljava da je $PR_{t,r_2} \geq T_r$. $PT_{r_1,t}$ označava snagu koju prima tag t od čitača r_1, a PR_{t,r_2} je snaga koji prima čitač r_2 od taga t, dok T_t i T_r označavaju pragove osetljivosti taga i čitača, respektivno. RS je skup čitača, a TS skup tagova u mrežnom domenu. $PT_{r_1,t}$ i PR_{t,r_2} se računaju primenom jednačina (5.11) i (5.13), respektivno.

$$P_t[dBm] = P_1[dBm] + G_r[dBi] + G_t[dBi] - L[dB], \tag{5.11}$$

gde je P_1 snaga koju emituje čitač, G_r i G_t su pojačanja signala antena čitača i tagova, respektivno, a L označava slabljenje signala koje se izračunava primenom Frisove formule:

$$L[dB] = 10log[(4\pi/\lambda)^2 d^n] + \delta[dB], \tag{5.12}$$

gde d označava fizičku razdaljinu između čitača i taga, n je faktor okruženja koji varira između 1.5 i 4 zbog promena u fizičkim uslovima, dok δ označava gubitke nastale zbog prirode bežičnih komunikacija.

Snaga koju čitač prima od taga računa se kao:

$$P_r[dBm] = P_b[dBm] + G_r[dBi] + G_t[dBi] - 20log(4\pi d/\lambda), \tag{5.13}$$

gde P_b predstavlja reflektovanu snagu koju šalje tag. Vrednost P_b zavisi od koefi-

cijenta refleksije taga Γ i od snage koju prima tag P_t (u vatima). P_b se računa na sledeći način:

$$P_b = (\Gamma_{tag})^2 P_t \tag{5.14}$$

Konačno, uzimajući sve u obzir, stopa pokrivenosti mreže se računa prema jednačini:

$$COV = \sum_{t \in TS}^{max} CV(t)/N_t \cdot 100\%, \tag{5.15}$$

gde se CV_t računa kao:

$$CV_t = \begin{cases} 1, & \text{ako } \exists\ r_1, r_2 \in RS,\ PT_{r_1,t} \geq T_t \wedge PR_{t,r_2} \geq T_r \\ 0, & \text{u suprotnom,} \end{cases} \tag{5.16}$$

gde je $N_t = |TS|$ broj tagova koji su distribuirani u domenu RFID mreže.

Osim navedene, postoje i druge formulacije cilja optimalne pokrivenosti tagova koja se definiše u zavisnosti od snage koju prima tag [141]. U ovom slučaju se optimalna pokrivenost formuliše kao suma razlike između željenog nivoa snage P_d i snage P_i^r koju prima svaki tag i [141]:

$$\min\ C = \sum_{i=1}^{N_T} (P_i^r - P_d), \tag{5.17}$$

gde je N_T broj tagova u domenu mreže, a P_d predstavlja vrednost praga osetljivosti.

Prva formulacija cilja optimalne pokrivenosti tagova je bolja zato što garantuje da će tag biti pokriven, ako postoje čitači koji zadovoljavaju određene uslove, nezavisno od praga osetljivosti. Za razliku od nje, druga formulacija je indirektna i ona se formuliše samo u zavisnosti od toga da li je snaga koju prima tag veća ili manja od praga osetljivosti.

Drugi po važnosti cilj problema planiranja RFID mreže je minimizacija broja čitača. Ovaj cilj je važan zbog činjenice da ukupni troškovi mreže zavise od ovog broja, koji matematički može da se formuliše kao [140]:

$$RS = N_{max} - N_{red}, \tag{5.18}$$

gde RS predstavlja skup čitača koji se nalaze na mreži, N_{max} je ukupan broj raspoloživih čitača i R_{red} je broj redudantnih čitača koje algoritam pronalazi.

Ni jedna ni druga hibridna metaheuristika ABC u modeliranju RFID problema ne koriste eksplicitno ovaj cilj.

RFID problem mrežnog planiranja je primer višekriterijumske optimizacije. Neki kriterijumi su protivrečni i postavlja se pitanje šta sa njima da se radi. Generalno, za probleme optimizacije sa više funkcija cilja, mogu da se koriste pristupi poput težinske sume ili prioritizacije ciljeva. Međutim, u ovom slučaju ni jedan takav pristup ne bi dao zadovoljavajuće rezultate, jer je cilj minimizacije broja čitača vrlo različit od ostalih. Drugi kriterijumi podležu klasičnoj optimizaciji, jer postoje vektori rešenja oko kojih se pretraga izvodi manje, ili više primenom eksploatacije i eksploracije.

S druge strane, broj čitača je jedan od parametara koji bi lako mogao da se inkorporira sa ostalim parametrima i algoritam bi se onda standardno izvršavao. Međutim, ovaj parametar ima sasvim drugačije osobine od ostalih. Na primer, dok se snaga čitača malo povećava ili smanjuje, ili se njegova pozicija pomera malo levo ili desno i na taj način se vrši optimizacija, sa brojem čitača takva optimizacija ne može da se radi na standardni način jer su u pitanju celobrojne vrednosti i kao takve su nezgodne za modifikovanje u ovom slučaju. Ako bi se koristila pretraga na osnovu broja čitača, svaka promena ovog parametra bi značila resetovanje algoritma i algoritam bi počinjao potragu iz početka. U takvim situacijama algoritam bi se ponašao oscilatorno, jer bi sa svakim resetovanjem gubio informacije o prethodno pronađenim rešenjima.

Zbog ovoga, GI-ABC za minimizaciju broja čitača koristi neki vid iscrpne pretrage [136], dok drugi ABC hibrid koristi jednostavnu heuristiku koja ispituje topologiju mreže, identifikuje klastere tagova i na osnovu ovih informacija određuje optimalan broj čitača [137].

Do interferencije dolazi kada više čitača istovremeno ispituje isti tag. Minimizacija interferencije čitača je važan cilj strategije implementacije RFID mreže:

$$INT = \sum_{t \in TS} \gamma(t), \tag{5.19}$$

gde je:

$$\gamma(t) = \sum PT_{r,t} - max\{PT_{r,t}\},\ r \in RS \wedge PT_{r,t} \geq T_t \tag{5.20}$$

Cilj minimizacije emitovane energije je najmanje važan cilj zato što je u neposrednoj konfrontaciji sa najvažnijim ciljem da se ostvari optimalna pokrivenost tagova (jednačine (5.11) i (5.13)). Ovaj cilj može da se modelira kao [140]:

$$SPOW = \sum_{r \in RS} PS_r, \tag{5.21}$$

gde PS_r označava snagu koju emituje čitač r.

5.1.4.2 Prilagođavanje hibridne metaheuristike

Iz pregleda literature vidi se da je problem planiranja RFID mreže sa više funkcija cilja već rešavan primenom metaheuristika inteligencije rojeva. PSO algoritam sa mehanizmom za eliminaciju redudantnih čitača je prikazan u [140], dok je opis kooperativnog ABC algoritam za optimizaciju sa više funkcija cilja (cooperative multi-objective artificial bee colony - CMOABC) dat u [138]. Osim navedenih, korišćeni su još i hijerarhijski ABC (eng. hierarchical artificial bee colony - HABC) [14], više-kolonijski algoritam bakterija (eng. multi-colony bacterial foraging optimization - MC-BFO) [142] i algoritam simulacije rasta biljaka (eng. plant growth simulation algorithm - PSGA) [143].

U prvoj hibridnoj implementaciji korišćen je GI-ABC [136], koji je detaljno opisan u Sekciji 5.1.3. Obzirom da se GI-ABC pokazao kao robusnija optimizaciona metoda od osnovnog ABC algoritma, pošlo se od predpostavke da ovaj pristup može dati bolje rezultate i u slučaju optimizacije problema planiranja RFID mreže.

Za optimizaciju kriterijuma minimizacija broja čitača, GI-ABC koristi vođenu iscrpnu pretragu (eng. guided exchaustive search). Na ovaj način se jednostavnije tretira ovaj cilj. Kako je već objašnjeno, broj čitača ne može da se tretira kao jedan od parametara algoritma, zato što ima bitno različita svojstva od ostalih parametara. Zbog toga se koristi skraćeni oblik iscrpne pretrage koji počinje sa 8 čitača. Dokle god je optimalna pokrivenost tagova na nivou od 100%, broj čitača se smanjuje za jedan, sve dok se ne dođe do minimalnog broja čitača sa kojom se drži potpuna pokrivenost tagova [136]. Za minimizaciju broja čitača, kao reper se uzima optimalna pokrivenost tagova, pošto je on važniji od svih drugih ciljeva.

Kada se jednom odredi optimalni broj čitača za određenu topologiju RFID mreže, nastavlja se klasična optimizacija GI-ABC algoritmom.

U drugom pristupu koji je predložen, hibridizovan je ABC sa heuristikom koja određuje broj i početnu lokaciju RFID čitača [137]. Planiranje RFID mreže je problem sa 4 parametra: x i y koordinatama čitača, emitovanom energijom čitača i brojem čitača. Međutim, kao što je objašnjeno, broj čitača ne može eksplicitno da se koristi kao parametar i zato se koristi hibridizacija sa heuristikom.

Hibridni algoritmi poboljšavaju sposobnost pretrage u odnosu na idividualne algoritme tako što efikasno kombinuju prednosti jednih sa nedostacima drugih algoritama. U ovom slučaju jednostavna heuristika pomoću prostih pravila pretrage može efikasno da odredi optimalan broj čitača za svaku topologiju RFID mreže.

Hibrid ABC i heuristike je dat u formi kolaboravnog hibrida sa sekvencijalnom strukturom. Hibrid prvo koristi heuristiku koja određuje broj i približnu lokaciju čitača tako što izračunava broj klastera tagova u datoj topologiji i tek onda izvodi optimizaciju primenom ABC jednačina pretrage. Dakle, kao ulaz u ABC algoritam se daje broj i približna lokacija RFID čitača, a ABC pretražuje prostor u okolini ovih lokacija optimizujući pritom pokrivenost tagova, interferenciju čitača i emitovanu snagu.

Za izračunavanje broja i približne lokacije klastera tagova koriste se relativno jednostavna heuristička pravila. Velika preciznost nije potrebna u ovom slučaju, pošto ABC algoritam sprovodi proces pretrage koji je dovoljno precizan.

Uzimajući u obzir fizičke karakteristike RFID sistema, koji je korišćen kao ilustracija u ovom primeru, čitač sa minimalnom snagom od 20 dBm (0.1W) može da prekrije radijus od 4.986 m, dok čitač sa najvećom snagom od 33 dBm (1.995W) prekriva radijus od 14.897 m. U Sekciji 5.1.4.3 dat je opis primera koji je uzet za potrebe testiranja. Za izračunavanje radijusa d, korišćena je formula:

$$d[m] = (\frac{10^{(P_r[dBm]+G_r[dBi]+G_t[dBi]-P_t[dBm]-\delta[dB])/10}}{(4\pi/\lambda)^2})^{\frac{1}{n}} \tag{5.22}$$

Pošto velika preciznost pretrage nije potrebna, za potencijalne lokacije čitača uzete su samo celobrojne tačke. Budući da se RFID mreža u korišćenom primeru prostire na 50 x 50m, prostor pretrage sastoji se iz $51 * 51$ tačaka (potencijalne lokacije čitača), što predstavlja relativno lak zadatak iscrpne pretrage. Kvalitet svake tačke može da se odredi u odnosu na centar klastera, koji može da se izračuna primenom različitih metoda. U ovom primeru [137], za svaku tačku su evaluirani čitači sa tri nivoa snage - maksimalnom, srednjom i minimalnom, što odgovara krugovima sa velikim, prosečnim i malim radijusom. Prvo se izračunava koncentracija tagova u malom krugu, a onda u srednjem (prsten oko malog kruga), itd. Za vrednost funkcije podobnosti svakog klastera, uzeta je razlika u broju tagova između srednjeg i malog kruga [137].

Na ovaj način se procenjuje broj i približna lokacija čitača. ABC algoritam uzima ovako izračunat broj čitača kao fiksnu vrednost, ali sprovodi pretragu u okolini približnih lokacija čitača koje je odredila heuristika.

U testiranjima oba algoritma korišćen je hijerarhijski pristup za upravljanje ciljevima. Prvo je optimizovana pokrivenost tagova, onda interferencija, i konačno emitovana energija.

5.1.4.3 Eksperimentalni rezultati

Oba algoritma su testirana na šest standardnih instanci problema RFID mrežnog planiranja sa više funkcija cilja iz literature: $C30$, $C50$, $C100$, $R30$, $R50$ i $R100$ sa 30,

50 i 100 tagova, raspoređenih u formi klastera i nasumično. Sve benčmark instance su preuzete sa javnog URL-a: http://www.ai.sysu.edu.cn/GYJ/RFID/TII/. Tagovi su distribuirani na prostoru od 50 x 50m. Algoritmi sa kojima je vršena komparativna analiza su testirani na istom setu podataka [140].

Za potrebe testiranja korišćen je ilustrativni primer [137], [136] sa različitim brojem čitača sa snagom koja se podešava u opsegu $[20, 33]dBm$ (0.1 do 2 W). Talasna dužina λ je postavljena na vrednost od 0.328m (915 MHz), pragovi osetljivosti tagova i čitača $T_t = -14\ dBm$ i $T_r = -80\ dBm$, respektivno, sa odgovarajućim pojačavanjem antena od $G_t = 3.7\ dBi$ i $G_r = 6.7\ dBi$. Vrednost δ je postavljena na 2, n na 2, a Γ_{tag} na 0.3.

Parametri ABC algoritma su postavljeni na sledećin način: veličina populacije N na 20, broj iteracija (MIN) na 20,000 sa ukupno 400,000 evaluacija funkcija. Isti broj evaluacija funkcija je korišćen i u [140]. MR je postavljen 0.8, a $limit$ na 1,000. Parametri koji su specifični za GI-ABC su podešeni na sledećin način: bp je postavljen na vrednost od 3,000, rr na 0.9, a mpr na 0.01.

Oba algoritma su upoređivana međusobno, ali i sa GPSO (global PSO), VPSO (von Neumann PSO) i GPSO i VPSO sa ugrađenim mehanizmom za eliminaciju redudantnih čitača - GPSO-RNP i VNPSO-RNP [140].

Eksperimenti su izvođeni u 50 nezavisnih puštanja algoritama sa različitim semenom pseudo-slučajnog broja. U Tabeli 5.3, gde su prikazani eksperimentalni rezultati, prikazane su najbolje i prosečne vrednosti za sve ciljeve.

Radi bolje preglednosti, u Tabeli 5.3 su najbolji rezultati iz svake katerogije označeni masnim slovima. Iz tabele se vidi da su obe hibridne metaheuristike predložene u [137] i [136], za većinu benčmark problema bolje od drugih algoritama tako što postižu mnogo bolju pokrivenost mreže sa manjim brojem čitača i manjom emitovanom energijom, dok interferencija postoji samo u problemima sa velikim brojem nasumično distribuiranih tagova. Zbog ovoga su ostvareni rezultati vrlo značajni. Kao napomena navodi se da je u Tabeli 5.3 prikazana izračena energija.

Algoritmi sa kojima je vršena komparativna analiza takođe postižu 100% pokrive-

Tabela 5.3: Eksperimentalni rezultati za šest RFID benčmark problema

Algoritam	Prosečne vrednosti				Najbolje vrednosti			
	Pokrivenost	Br.čit.	Interf.	Snaga	Pokrivenost	Br.čit.	Interf.	Snaga
Rezultati za benčmark C30								
GPSO	100.00 %	6	0.000	35.074	100.00 %	6	0.000	31.865
VNPSO	100.00 %	6	0.000	34.762	100.00 %	6	0.000	31.951
GPSO-RNP	100.00 %	3.18	0.000	35.511	100.00 %	3	0.000	33.948
VNPSO-RNP	100.00 %	3.04	0.000	35.034	100.00 %	3	0.000	33.535
VKP	100.00 %	**2**	0.000	16.246	100.00 %	**2**	0.000	15.297
GI-VKP	100.00 %	**2**	0.000	**15.739**	100.00 %	**2**	0.000	**15.050**
Rezultati za benčmark C50								
GPSO	95.60 %	6	0.000	35.170	100.00 %	6	0.000	31.852
VNPSO	99.20 %	6	0.000	35.023	100.00 %	6	0.000	31.742
GPSO-RNP	100.00 %	5.04	0.000	36.244	100.00 %	5	0.000	33.418
VNPSO-RNP	100.00 %	5.06	0.000	36.565	100.00 %	5	0.000	34.522
VKP	100.00 %	**4**	0.000	26.673	100.00 %	**4**	0.000	23.800
GI-VKP	100.00 %	**4**	0.000	**26.271**	100.00 %	**4**	0.000	**23.320**
Rezultati za benčmark C100								
GPSO	98.34 %	6	0.002	38.652	100.00 %	6	0.000	37.374
VNPSO	99.72 %	6	0.000	38.167	100.00 %	6	0.000	36.803
GPSO-RNP	100.00 %	5.16	0.000	38.800	100.00 %	5	0.000	37.513
VNPSO-RNP	100.00 %	5.04	0.000	38.513	100.00 %	5	0.000	37.449
VKP	100.00 %	**4**	0.000	33.638	100.00 %	**4**	0.000	30.066
GI-VKP	100.00 %	**4**	0.000	**33.113**	100.00 %	**4**	0.000	**29.851**
Rezultati za benčmark R30								
GPSO	92.13 %	6	0.000	38.849	100.00 %	6	0.000	38.842
VNPSO	94.53 %	6	0.000	38.849	100.00 %	6	0.000	38.655
GPSO-RNP	99.87 %	7.46	0.002	39.821	100.00 %	6	0.000	39.265
VNPSO-RNP	100.00 %	6.86	0.003	40.143	100.00 %	6	0.000	39.574
VKP	100.00 %	**5**	0.000	37.475	100.00 %	**5**	0.000	32.747
GI-VKP	100.00 %	**5**	0.000	**36.961**	100.00 %	**5**	0.000	**32.551**
Results for benchmark R50								
GPSO	92.52 %	6	0.000	39.692	98.00 %	6	0.000	40.520
VNPSO	93.96 %	6	0.000	39.690	98.00 %	6	0.000	39.595
GPSO-RNP	99.84 %	8.26	0.012	40.652	100.00 %	7	0.000	40.315
VNPSO-RNP	100.00 %	7.66	0.030	40.667	100.00 %	7	0.000	40.080
VKP	100.00 %	**5**	0.006	41.273	100.00 %	**5**	0.000	39.017
GI-VKP	100.00 %	**5**	**0.005**	**41.132**	100.00 %	**5**	0.000	**38.900**
Rezultati za benčmark R100								
GPSO	91.18 %	6	0.014	40.074	95.00 %	6	0.000	40.098
VNPSO	94.14 %	6	0.012	40.333	97.00 %	6	0.043	40.657
GPSO-RNP	99.74 %	9.24	0.118	41.505	100.00 %	8	0.000	40.925
VNPSO-RNP	100.00 %	8.44	0.242	41.462	100.00 %	8	0.000	41.031
VKP	100.00 %	**5**	0.015	44.721	100.00 %	**5**	0.006	40.011
GI-VKP	100.00 %	**5**	0.015	**44.188**	100.00 %	**5**	0.006	**39.961**

nost mreže u većini test slučajeva, ali uz veći broj čitača i veću izračenu energiju [140]. Kao zaključak se navodi da su obe predložene hibridne metaheuristike bolje od drugih algoritama u rešavanju ovog problema.

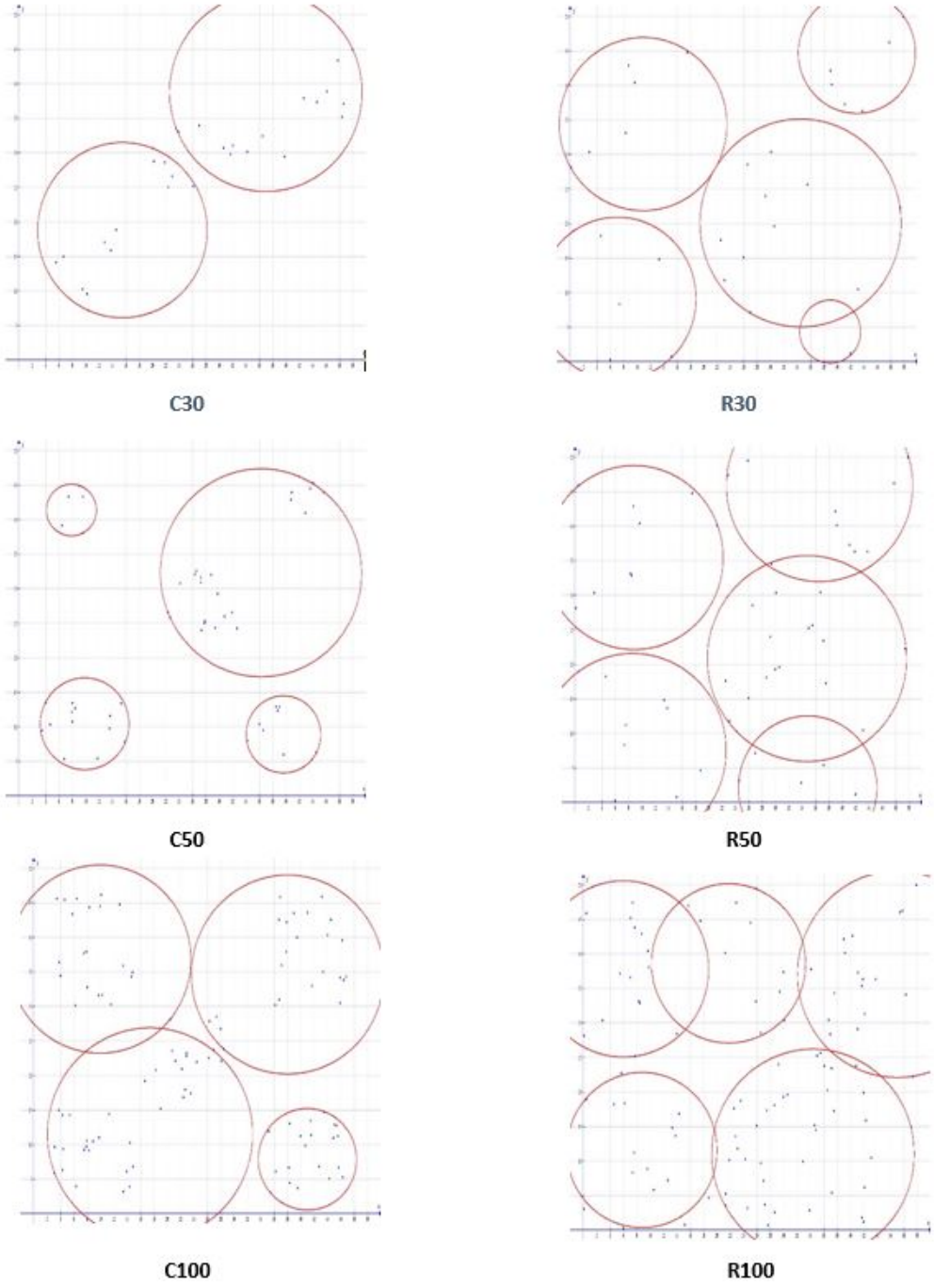
Kada se uporede ABC sa heuristikom i GI-ABC vidi se da pojačana intenzifikacija u kasnijim iteracijama omogućava generisanje boljih rezultata. Za sve testove, i u slučaju najboljih i u slučaju prosečnih vrednosti, emitovana energija je bolja kod metaheuristike GI-ABC. I u ovom slučaju vidi se da se inkorporiranjem genetskih operatora postiže bolji balans između eksploatacije i eksploracije, a brzina konvergencije se poboljšava.

Na Slici 5.1, prikazani su optimalni rezultati za skup benčmark problema gde su tagovi grupisani u klasterima (C30,C50 i C100) i nasumično (R30, R50 i R100).

Sa slike se vidi da su predloženi algoritmi efikasni. Krugovi prolaze kroz dve ili tri tačke, što znači da je snaga maksimalno smanjena. Sa slike se takođe vidi da su čitači logično raspoređeni i da je u svakom slučaju ostvarena stopostotna pokrivenost svih tagova.

Na primer, kod C30 problema, vidi se da ima pet klastera tagova, ali se koriste samo dva čitača sa jačom snagom, jer je tako efikasnije. U slučaju C50, gde ima više klastera tagova koji su razbacani, algoritam je generisao četiri čitača, od čega jedan čitač ima više snage, a preostala tri imaju značajno manje snage. Na ovaj način je snaga minimizovna, pošto je jači čitač pokrio veliki broj tagova. Za benčmark C100 generisano je četiri čitača, i to tri sa relativno velikom snagom i jedan sa manjom u gonjem desnom uglu topologije.

U slučaju R30, napravljena su pet čitača. Jedan mali čitač nalazi se u donjem desnom uglu, pošto je tamo lociran jedan usamljeni tag i algoritam je morao pomoću posebnog čitača da pokrije taj tag. Na primeru R50, algoritam je takođe generisao pet čitača. Bez obzira što se domet nekih čitača preklapa, interferencija ne postoji, pošto se u njihovom preseku ne nalazi ni jedan tag. Konačno, u benčmarku R100, gde ima mnogo nasumično razbacanih tagova, algoritam je uspeo da optimizuje takođe sa pet čitača. Međutim, u ovom slučaju primećuje se blaga interferencija.



Slika 5.1: Optimalni položaj čitača dobijen testiranim algoritmom

5.1.5 Zaključak

ABC je jedna od najpopularnijih metaheuristika rojeva koju je razvio Karaboga za probleme globalne optimizacije bez ograničenja [51], a zvaničnije su je predstavili Karaboga i Bastuk [52], [53]. Uvidom u pregledne radove, ustanovljeno je da je ABC

efikasna metaheuristika koje je uspešno primenjivana na veliki broj eksperimentalnih i praktičnih problema [78], [79].

Međutim, teorijskom analizom i empirijskim testovima uočeno je da metaheuristika ABC pokazuje nedostatke. Kao prvo, utvrđeno je da proces intenzifikacije nije dovoljno jak, što za posledicu ima slabu konvergenciju ka optimalnom domenu prostora pretrage. Jednačina pretrage ABC algoritma (5.2) koristi mutaciju, bez eksplicitne primene operatora ukrštanja, što utiče na slabiju intenzifikaciju. Takođe, u ovoj jednačini, procesi intenzifikacije i diversifikacije nisu dovoljno razdvojeni, pa se ne zna tačno kada algoritam izvodi diversifikaciju, a kada izvodi intenzifikaciju.

Nedostatak u vidu slabe intenzifikacije vodi ka problemu loše uspostavljenog balansa između eksploatacije i eksploracije. Problem lošeg balansa je izraženiji u ranijim iteracijama, pošto u ovoj fazi jednačina intenzifikacije (5.2) češće izvodi diversifikaciju. Međutim, ovaj problem je ozbiljniji u kasnijim iteracijama kada je potrebno da se vrši fina pretraga optimalnog domena.

Pokazano je da problemi spore konvergencije i lošeg balansa eksploatacije i eksploracije mogu da se prevaziđu hibridizacijama. U prvoj implementaciji prikazan je GI-ABC hibrid niskog nivoa koji integriše genetske operatore ukrštanja i mutacije u osnovnu metaheuristiku ABC za probleme sa ograničenjima [130]. Ovaj hibrid rešava problem lošeg balansa između intenzifikacije i diversifikacije u kasnijim iteracijama. Predpostavka je da je algoritam u ranijim ciklusima pronašao optimalni domen i da je u kasnijim ciklusima potrebno pojačati intenzifikaciju i simultano smanjiti diversifikaciju, kako bi se vršila fina pretraga oko globalnog optimuma.

GI-ABC u kasnijim iteracijama zamenjuje mehanizam pčele izviđača, koji izvodi diversifikaciju, sa mehanizmom vođenog posmatrača, koji koristi snažnu intenzifikaciju primenom uniformnog ukrštanja i mutacije. Vođeni posmatrač kombinuje najbolja rešenja u populaciji i na taj način vrši finu pretragu oko optimalnog rešenja.

GI-ABC je testiran na standardnim benčmark problemima globalne optimizacije sa ograničenjima. Iz komparativne analize sa osnovnom metaheuristikom ABC koja je primenjena na iste probleme, zaključuje se da GI-ABC ispravlja navedeni nedostatak originalnog algoritma. Takođe, na osnovu testova sa drugim metaheuristikama,

zaključuje se da GI-ABC spada u vrhunske optimizacione metode.

U drugoj implementaciji prikazan je ABC sa pretragom svitaca (eng. ABC - firefly search - ABC-FS), koji je prilagođen za praktičan problem optimizacije portfolija sa ograničenjima kardinalnosti [144]. Iako su benčmark problemi dobri mehanizmi za merenje performansi algoritama, radi bolje analize potrebno je da se ustanovi kako se algoritmi ponašaju i na praktičnim problemima iz svakodnenog života.

ABC-FS rešava probleme slabije intenzifikacije osnovnog ABC algoritma i balansa između eksploatacije i eksploracije. Pošto FA jednačina pretrage ispoljava jaku intenzifikaciju, pošlo se od predpostavke da kombinacija ova dva pristupa može eliminisati nedostatke ABC algoritma. FA pretraga je integrisana u metodu pčele radilice, tako što se u nekim ciklusima koristi standardna pretraga pčele radilice, a u nekim FA pretraga. Empirijskim testovima je potvrđeno da ovaj hibridni pristup ispravlja nedostatke osnovne metaheuristike ABC, o čemu je više detalja navedeno u narednoj sekciji.

U trećoj implementaciji, GI-ABC je prilagođen za rešavanje praktičnog problema planiranja RFID mreže [136]. Problem planiranja RFID mreže spada u grupu teških problema globalne optimizacije sa više funkcije cilja. U implementaciji metaheuristike GI-ABC za ovaj problem, korišćen je princip hijerarhijskog upravljanja ciljevima, gde se prvo optimizuje prvi cilj po važnosti, zatim drugi, treći, itd.

GI-ABC za problem planiranja RFID mreže je upoređivan sa drugim vrhunskim metaheuristikama, ali i sa jednom varijantom osnovnog ABC algoritma koja je hibridizovana sa heuristikom za određivanje broja i početnih lokacija čitača [137]. Ova hibridizacija ne menja način funkcionisanja ABC algoritma, pošto se izlazni podaci heuristike koriste kao ulaz u metaheuristiku ABC. GI-ABC je pokazao mnogo bolje vrednosti za optimizaciju po kriterijumu emitovane energije, na osnovnu čega se najbolje vidi da je brzina konvergencije osnovnog algoritma poboljšanja. Takođe, u odnosu na druge metaheuristike, GI-ABC je postigao značajno bolje rezultate.

Kao zaključak se navodi da je moguće ispraviti nedostatke osnovne metaheuristike ABC tako što se navedeni nedostaci ispravljaju pažljivom hibridizacijom sa strukturalnim elementima drugih metaheuristika koji pokazuju snagu u domenima gde je

ABC slab.

5.2 Hibridizacija metaheuristike FA

5.2.1 Osnovna metaheuristika FA

Svetlosna signalizacija svitaca poslužila je kao inspiracija za nastanak metaheuristike FA, koja pripada grupi novijih algoritama rojeva. Tvorac ovog algoritma je Yang, a u prvoj prikazanoj implementaciji, algoritam je adaptiran za optimizaciju neprekidnih problema bez ograničenja [54].

U ovom slučaju su za inspiraciju iskorišćeni neki aspekti ponašanja svitaca. Svici su interesantni insekti koji sami proizvode svetlost i na osnovnu tog svetla prilagođavaju svoje međusobno ponašanje. Neki kažu da je svetlost signal za parenje i neka vrsta mehanizma za upozoravanje koji štiti jata svitaca od potencijalnih napada grabljivica.

Treptuće svetlo svitaca se formuliše na način da može da se napravi analogija sa funkcijom cilja koja se optimizuje, što omogućava kreiranje novih optimizacionih algoritama [44]. Da bi se modeliralo ponašanje svitaca u prirodi, FA sledi tri pravila koja uprošćavaju realni model [54]:

- svaki svitac privlači sve druge svice slabijeg intenziteta svetlosti, pri čemu se pol svitaca zanemaruje;
- privlačnost između svitaca je direktno proporcionalna sa emitovanim intenzitetom svetlosti, dok je s druge strane jačina svetlosti obrnuto proporcionalna udaljenosti svica od izvora svetlosti i
- intenzitet svetlosti svica je u najmanju ruku pod uticajem, ili potpuno određen vrednošću distribucije funkcije cilja.

U implementaciji metaheuristike FA se postavljaju dva bitna pitanja: varijacija intenziteta svetlosti i formulacija atraktivnosti (način na koji jedan svitac u populaciji privlači drugog svica). Zbog jednostavnosti, može da se predpostavi da je nivo privlačnosti određenog svica određen intenzitetom njegove svetlosti, koja dalje zavisi

od vrednosti kodirane funkcije cilja [44]. U najjednostavnijem slučaju problema maksimizacije, osvetljenost I određenog svica na lokaciji x može da se odredi kao $I(x) \approx f(x)$. Međutim, s obzirom da atraktivnost svica β može da se definiše i s aspekta posmatrača i s aspekta drugih svitaca, ona predstavlja relativnu veličinu i varira u zavisnosti od udaljenosti r_{ij} između svitaca i i j. Osim toga, prema zakonima fizike, intenzitet svetlosti se smanjuje kako se posmatrač udaljava od njenog izvora i medijum kao što je vazduh apsorbuje deo intenziteta svetlosti kako svetlost putuje od izvora ka odredištu. Zbog ovoga i atraktivnost svitaca u velikoj meri zavisi od ovih parametara.

Sa povećanjem udaljenosti posmatrača od izvora svetlosti, intenzitet svetlosti se smanjuje sa kvadratom razdaljine između izvora i posmatrača [54]:

$$I(r) = \frac{I_s}{r^2}, \tag{5.23}$$

gde I_s označava intenzitet svetlosti na izvoru, a r udaljenost između izvora svetlosti i posmatrača. Za medijum kroz koji prolazi svetlost sa koeficijentom apsorpcije γ, intenzitet svetlosti I se menja po udaljenosti r na osnovu sledeće jednačine:

$$I = I_0 e^{-\gamma r}, \tag{5.24}$$

gde je I_0 prvobitni intenzitet svetlosti. Da bi se predupredio slučaj da imenilac bude jednak 0 u jednačini (5.23), za slučaj kada je $r = 0$, kombinovani efekat obrnutog kvadratnog zakona i apsorpcije medijuma se prikazuje u sledećom Gausovom obliku:

$$I(r) = I_0 e^{-\gamma r^2} \tag{5.25}$$

U skladu sa prethodnim jednačinama, atraktivnost β svitaca se određuje kao:

$$\beta = \beta_0 e^{-\gamma r^2}, \tag{5.26}$$

gde je β_0 atraktivnost u tački $r = 0$.

Obzirom da je računanje eksponencijalne funkcije zahtevno u smislu korišćenja računarskih resursa, jednačina (5.26) se u većini FA implementacija zamenjuje sledećim izrazom:

$$\beta = \frac{\beta_0}{1 + \gamma r^2} \tag{5.27}$$

Udaljenost između bilo koja dva svica i i j na pozicijama x_i i x_j, respektivno računa se pomoću Kartezijanske razdaljine [54]:

$$r_{ij} = ||x_i - x_j|| = \sqrt{\sum_{k=1}^{D}(x_{i,k} - x_{j,k})^2}, \tag{5.28}$$

gde $x_{i,k}$ predstavlja k-tu komponentu prostorne koordinate x_i, i-tog svica, a D označava broj varijabli optimizacionog problema.

Konačno, kretanje svitca, tj. proces ispitivanja prostora pretrage, može da se definiše u formi sledeće jednačine [44]:

$$x_i = x_i + \beta_0 e^{-\gamma r_{i,j}^2}(x_j - x_i) + \alpha\epsilon_i, \tag{5.29}$$

gde prvi sabirak predstavlja trenutnu poziciju svica, drugi definiše varijaciju privlačnosti, a treći sabirak uvodi slučajnost pomoću parametra α koji se naziva parametar slučajnosti, dok je ϵ_i vektor pseudo-slučajnih brojeva unifromno distribuiranih u intervalu između 0 i 1. U većini implementacija uzima se da je $\beta_0 = 1$, a da $\alpha \in [0, 1]$.

Parametar γ, koji modelira varijacije privlačnosti, ima presudan uticaj na brzinu konvergencije algoritma i način na koji se metaheuristika FA ponaša. U teoriji je $\gamma \in [0, \infty]$, ali se u praksi $\gamma = O(1)$ određuje na osnovu intervala u kome se nalaze vrednosti varijabli problema koji se optimizuje. U većini aplikacija, ovaj parametar varira u opsegu od 0.1 do 10 [44].

Psuedo-kod osnovne metaheuristike FA prikazan je u Algoritmu 11. U navedenom pseudo-kodu, FN označava veličinu populacije, IN je ukupan broj iteracija u jednom izvršavanju algoritma, a t je brojač trenutne iteracije.

Algoritam 11 Pseudo-kod osnovnog FA

```
početak
generiši početnu populaciju svitaca X_i, i = 1, 2, 3, ..., FN
intenzitet svetlosti svica I_i u tački x_i se određuje pomoću f(x)
odredi broj iteracija algoritma IN
t = 0
while t < IN do
   for i = 1 do (FN − 1) do
      for j = (i + 1) do FN do
         if I_j < I_i then
            pomeri svica j ka svicu i u d dimenzija
            izmeri podobnost novog rešenja i zameni staro novim rešenjem ako je
            podobnost novog veća i ažuriraj intenzitet svetlosti
         end if
      end for
   end for
   sortiraj sve svice, zapamti najboljeg i pomeri ih na slučajne pozicije
   t++
end while
kraj
```

FA je primenjivan na veliki broj benčmark i praktičnih problema globalne optimizacije. Tako na primer, osim prve implementacije Yang-a [54], Lukasik i ostali su uspešno implementirali metaheuristiku FA za rešavanje test problema neprekidne optimizacije sa ograničenjima vrednosti varijabli [71]. Yang je dao sveobuhvatni pregled primena algoritma FA za rešavanje problema globalne optimizacije [145].

5.2.2 Nedostaci metaheuristike FA

Analizom osnovne jednačine pretrage metaheuristike FA (5.29) vidi se da se operator mutacije koristi i za lokalnu i za globalnu pretragu. Kada se parametar ϵ_i generiše na osnovu Gausove distribucije ili Lévy leta i kada parametar α ima veću vrednost, intenzitet mutacije je veći (jača diversifikacija). S druge strane, ako parametar α ima malu vrednost, onda je i mutacija manja i ograničava se samo na lokalni domen prostora pretrage (slabija diversifikacija). Zbog efikasnosti operatora mutacije,

nedostatak eksplicitnog ukrštanja se ne odražava na proces intenzifikacije.

Da bi FA uspostavio bolji odnos između intenzifikacije i diversifikacije, dobra je praksa da se koristi dinamičko podešavanje parametra α. U početnim fazama izvršavanja algoritma, ovaj parametar ima veću vrednost, koja se zatim u svakoj iteraciji postepeno smanjuje. U osnovnoj FA jednačini pretrage ne postoji eksplicitna selekcija. Mehanizmi selekcije i rangiranja se izvršavaju u fazi ažuriranja rešenja.

Jedna od karakteristika FA koja razlikuje ovu metaheuristiku roja od ostalih je upotreba atraktivnosti (privlačnosti) kao mere podobnosti rešenja. S obzirom da je lokalna privlačnost jača nego privlačnost između svitaca koji su udaljeni jedan od drugog, populacija FA se deli u podgrupe, gde svaka grupa vrši pretragu lokalnog dela domena. U nekom od tih lokalnih domena nalazi se i globalno najbolje rešenje.

Osnovni FA daje dobre rezultate za probleme globalne optimizacije bez ograničenja [54]. Međutim, kada se primeni na probleme sa ograničenjima, ovaj algoritam postiže lošije rezultate na nekim problemima. FA nema jasno naglašenu jednačinu diversifikacije, a s obzirom da ograničenja značajno sužavaju prostor pretrage, često se dešava da se u ranijim ciklusima algoritam zaglavi u suboptimalnom domenu prostora pretrage. Za probleme sa ograničenjima, zapaža se da intenzitet diversifikacije u ranijim iteracijama često nije dovoljan, pa samim tim ni balans između diversifikacije i intenzifikacije nije sasvim adekvatan i više "vuče" na stranu intenzifikacije.

Prethodno zapažanje je izveden na osnovu testiranja osnovnog FA na standardnim G benčmark problemima globalne optimizacije sa ograničenjima. Parametri algoritma su pažljivo podešavani i rezultati za optimalnu kombinaciju vrednosti parametara su prikazani u Tabeli 5.4 [146].

Iz Tabele 5.4, vidi se da FA postiže odlične rezultate u testovima $G3$, $G8$, $G9$, $G11$ i $G12$, a u $G5$, $G6$, $G7$ i $G13$ postiže sasvim zadovoljavajuće performanse koje mogu da se porede sa rezultatima drugih vrhunskih metaheuristika.

Međutim, kada se analiziraju rezultati za benčmark funkcije $G1$, $G4$ i $G10$, zaključuje se da FA postiže lošije performanse i to za sva tri indikatora - najbolje, prosečne vrednosti i standardnu devijaciju. Analizom brzine konvergencije ustanovljeno je da

Tabela 5.4: Rezultati osnovnog FA za globalne probleme sa ograničenjima [146]

Problem	Optimum		FA
G1	-15.000	Najbolji	-12.432
		Prosečni	-10.161
		StDev	1.902
G2	-0.803619	Najbolji	-0.803619
		Prosečni	-0.782942
		StDev	0.022
G3	-1.000	Najbolji	-1.000
		Prosečni	-1.000
		StDev	0.000
G4	-30665.539	Najbolji	-30378.381
		Prosečni	-30259.518
		StDev	33.888
G5	5126.497	Najbolji	5126.497
		Prosečni	5126.835
		StDev	0.065
G6	-6961.814	Najbolji	-6961.811
		Prosečni	-6961.786
		StDev	0.011
G7	24.306	Najbolji	24.307
		Prosečni	24.311
		StDev	0.012
G8	-0.095825	Najbolji	-0.095825
		Prosečni	-0.095825
		StDev	0.000
G9	680.63	Najbolji	680.630
		Prosečni	680.630
		StDev	0.000
G10	7049.25	Najbolji	7072.411
		Prosečni	7388.856
		StDev	216.552
G11	0.75	Najbolji	0.750
		Prosečni	0.750
		StDev	0.000
G12	-1.000	Najbolji	-1.000
		Prosečni	-1.000
		StDev	0.000
G13	0.053950	Najbolji	0.054
		Prosečni	0.252
		StDev	0.102

u ovim slučajevima FA ne uspeva da pronađe optimalni domen prostora pretrage, pa su zato i najbolji i prosečni rezultati lošiji.

U slučaju testa $G2$, iako FA uspeva da pronađe optimum, prosečne vrednosti i standardna devijacija bi mogle da se poboljšaju. U nekim izvršavanjima, algoritam ima "sreće" i bez obzira na slabu diversifikaciju, uspeva da pogodi optimalni region i generiše optimalno rešenje. Međutim, u ostalim izvršavanjima, algoritam "promašuje", što ima neposredne implikacije na slabije prosečne vrednosti i slabiju standardnu devijaciju.

5.2.3 Hibridizacija FA sa metaheuristikom ABC

Na osobinu metaheuristike FA koja se manifestuje manje izraženom diversifikacijom može da se utiče hibridizacijom. Mehanizam pčele izviđača ABC algoritma, koji izvodi proces eksploracije, se i u eksperimentalnim i u praktičnim implementacijama pokazao kao veoma efikasan mehanizam za pronalaženje pravog dela prostora pretrage. Inkorporiranjem ovog mehanizma u metaheuristiku FA, snaga diversifikacije može značajno da se poboljša.

U toku istraživanja predložena su i implementirana su dva hibrida metaheuristika FA i ABC. Prvi pristup prilagođen je za rešavanje benčmark problema globalne optimizacije sa ograničenjima, dok je drugi adaptiran za jedan praktičan problem. Drugi pristup detaljnije je opisan s obzirom na neophodne modifikacije za praktičnu primenu.

Prvi pristup, koji je nazvan unapređeni FA (eng. enhanced FA - eFA), bavi se problemom neusklađenog balansa između intenzifikacije i diversifikacije, tako što uvodi mehanizam koji je analogan proceduri pčela izviđača ABC metaheuristike. Opis i rezultati ovog algoritma prikazani su u radu koji je prihvaćen za objavljivanje u jednom međunarodnom časopisu.

U početnim iteracijama izvršavanja metaheuristike eFA, više ciklusa se troši na eksploraciju, da bi se kasnije pojačala eksploatacija. Diversifikacija se pojačava uvođenjem tzv. iscrpljenog svica. Svaki svitac u populaciji koji ne može da se poboljša

u toku predefinisanog broja ciklusa dobija atribut „iscrpljen". Ovaj broj se kontroliše pomoću dodatnog kontrolnog parametra granica iscrpljenosti (eng. exhaustiveness limit - *EL*). *EL* je analogan parametru *limit* metaheuristike ABC. „Iscrpljeni" svici zamenjuju se sa slučajnim agentima.

Posle određenog broja ciklusa, predpostavlja se da je eFA pronašao optimalan domen prostora pretrage. U ovoj fazi više nema potrebe za intenzivnom eksploracijom koja se zamenjuje usmerenom intenzifikacijom oko najboljih jedinki u populaciji. Vremenski trenutak u izvršavanju algoritma nakon koga se mehanizam diversifikacije zamenjuje jakom intenzifikacijom naziva se tačka zamene (eng. replacement point - *RP*). Nakon *RP* ciklusa, svi „iscrpljeni" svici zamenjuju se hibridima koji se generišu ukrštanjem najboljih jedinki u populaciji primenom operatora uniformnog ukrštanja.

Analizom je ustanovljeno da, ako se intenzivna eksploatacija koristi u svakoj iteraciji algoritma, da će poplacija da konvergira previše brzo. U slučaju da algoritam nije pronašao optimalni domen u ranijim ciklusima, ova konvergencija će prouzrokovati lošije prosečne rezultate. Za rešavanje ovog problema, predloženo je uvođenje dodatnog kontrolnog parametra intenzitet eksploatacije (eng. exploitation intensity - *EI*). Primenom *EI* parametra, kontroliše se izvođenje intenzifikacije nakon *RP* ciklusa, tako što se u nekim iteracijama koristi diversifikacija, koja generiše slučajna rešenja, umesto intenzivne eksploatacije.

Drugi predloženi pristup koji kombinuje metaheuristike FA i ABC je nazvan modifikovani FA (eng. modified FA - mFA) [147] i on spada u grupu hibrida niskog nivoa, gde je mehanizam pčele izviđača koji izvodi snažnu diversifikaciju ugrađen u FA. Unapređenja osnovnog FA se pre svega ogledaju u tome što je diversifikacija pojačana u ranim fazama izvršavanja algoritma, da bi bila eliminisana u kasnijim fazama, kada je algoritam već konvergirao ka optimalnom domenu prostora pretrage. U cilju kontrolisanja izvršavanja diversifikacije, u algoritam je inkorporiran dodatni kontrolni parametar.

Hibridni pristup je adaptiran za rešavanje problema optimizacije portfolija prosečne varijanse sa ograničenjima kardinalosti i entropije (eng. cardinality constrained mean variance (CCMV) portfolio problem with entropy constraint), u nastavku

CCMV problem sa ograničenjem entropije [147]. Ovaj problem pripada grupi kombinovanih problema kvadratnog i celobrojnog programiranja i kao takav predstavlja dobar indikator efikasnosti hibridne metaheuristike.

5.2.3.1 Optimizacioni problem

Portfolio problem CCMV sa ograničenjem entropije je izveden iz standardnog modela efikasne granice i Markowitc-ovog modela. Model je inspirisan više-ciljnim pristupom Usta i Kantara koji se bazira na modelu selekcije portfolija prosečne varijanse i asimetrične entropije (eng. mean-variance-skewness portfolio model - MVSM) koji koristi Šanonovu entropijsku meru za generisanje diversifikovanog portfolija [148], [149]. Osnovnom CCMV portfolio modelu je dodata entropija da bi se generisao diversifikovani portfolio.

Model problema CCMV sa ograničenjem entropije može da se definiše pomoću sledećih jednačina [150]:

$$\min \ \lambda[\sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{i,j}] - (1-\lambda)[\sum_{i=1}^{N} x_i \mu_i] \tag{5.30}$$

u odnosu na ograničenja:

$$\sum_{i=1}^{N} x_i = 1 \tag{5.31}$$

$$\sum_{i=1}^{N} z_i = K \tag{5.32}$$

$$\varepsilon_i z_i \leq x_i \leq \delta_i z_i, \ z \in \{0,1\}, \ i = 1,2,3,...N \tag{5.33}$$

$$-\sum_{i=1}^{N} z_i x_i \ln x_i \geq L_E, \tag{5.34}$$

U jednačinama (5.30) - (5.34), N je broj potencijalnih akcija koje se mogu uključiti u portfolio, λ je parametar izbegavanja rizika, x_i i x_j su težinski koeficijenti akcija i i

j respektivno, a $\delta_{i,j}$ je njihova kovarijansa, i μ_i je prinos i-te akcije. K je željeni broj akcija koje će biti uključene u portfolio. Varijabla odlučivanja z_i određuje da li će akcija i biti uključena u portfolio. Ako je njena vrednost 1, akcija i će biti uključena, a ako je vrednost 0, akcija i se neće nalaziti u portfoliju. ε i δ predstavljaju donju i gornju granicu težinskih koeficijenata akcija koje se nalaze u portfoliju.

Za potrebe testiranja, u model je inkorporirano i ograničenje entropije (5.34) sa donjim granicama prikazanim u jednačini (5.35), da bi se osiguralo da diversifikacija portfolija nije previše mala. L_E je donja granica entropije iz opsega $[0, \ln K]$. U jednačini (5.34), z_i osigurava da se uzimaju u obzir samo one akcije koje su uključene u portfolio.

Ograničenje entropije definiše donju granicu L_E entropije $E(P)$ portfolija P prema sledećoj jednačini [151]:

$$E(P) = -\sum_{i=1}^{N} x_i \ln x_i \geq L_E, \tag{5.35}$$

gde je N broj akcija u portfoliju P, a x_i je težinski koeficijent akcije i.

Iz formulacije navedenog modela optimizacije portfolija sa ograničenjem entropije, vidi se da model pripada grupi kombinovanih problema kvadratnog i celobrojnog programiranja. U njegovoj formulaciji se koriste i realne i celobrojne varijable sa ograničenjima jednakosti i nejednakosti i kao takav pripada grupi NP-teških problema.

5.2.3.2 Opis metaheuristike mFA

U fazi inicijalizacije mFA generiše slučajnu populaciju od SN svitaca pomoću jednačine [147]:

$$x_{i,j} = lb_j + rand(0,1) * (ub_j - lb_j), \tag{5.36}$$

gde $x_{i,j}$ predstavlja težinski koeficijent j-te akcije u portfoliju i-tog agenta u populaciji, $rand(0,1)$ je pseudo-slučajan broj uniformno distribuiran između 0 i 1, a ub_j

i lb_j su gornja i donja granica težinskih koeficijenata j-te akcije repsektivno.

Ako vrednost slučajno generisanog težinskog koeficijenta j-te akcije i-tog svica u populaciji ispadne iz opsega $[lb_j, ub_j]$, ona se vraća u predefinisane granice primenom sledećeg izraza:

$$\begin{aligned} &\text{ako } (x_{i,j}) > ub_j, \text{ onda je } x_{i,j} = ub_j \\ &\text{ako } (x_{i,j}) < lb_j, \text{ onda je } x_{i,j} = lb_j \end{aligned} \tag{5.37}$$

U fazi inicijalizacije se dodeljuju vrednosti varijabli odlučivanja $z_{i,j}$ $(i = 1, ...SN, j = 1, ...N)$ svakog agenta i, gde je N broj potencijalnih akcija u portfoliju. Dakle, svaki svitac u populaciji se modelira pomoću $2 * N$ dimenzija. z_i je binarni vektor čija je vrednost 1 ako je akcija uključena u portfolio, a 0 ako nije.

Vrednosti varijabli odlučivanja se određuju slučajno pomoću sledeće jednačine:

$$z_{i,j} = \begin{cases} 1, \text{ ako je } \phi < 0.5 \\ 0, \text{ ako je } \phi \geq 0.5 \end{cases} \tag{5.38}$$

gde je ϕ pseudo-slučajan realan broj između 0 i 1.

Da bi se garantovala dopustivost rešenja, koristi se algoritam dogovora (eng. arrangement algorithm), koji je predložen u [152]. U mFA implementaciji ovaj algoritam se primenjuje prvi put u fazi inicijalizacije.

U algoritmu dogovora, i je trenutno rešenje koje se sastoji iz: skupa Q različitih K_i^* akcija u i-tom rešenju, $z_{i,j}$ je varijabla odlučivanja akcije j i $x_{i,j}$ je težinski koeficijent akcije j. Pseudo-kod algoritma dogovora dat je u Algoritmu 12.

Algoritam 12 Pseudo-kod algoritma dogovora

while $K_i^* < K$ **do**
  izaberi slučajnu akciju j takvu da $j \notin Q$
  $z_{i,j} = 1, Q = Q \cup [j], K_i^* = K_i^* + 1$
end while
while $K_i^* > K$ **do**
  izaberi slučajnu akciju j takvu da $j \in Q$
  $z_{i,j} = 1, Q = Q - [j], K_i^* = K_i^* - 1$
end while
while true **do**
  $\theta = \sum_{j \in Q} x_{i,j}, x_{i,j} = x_{i,j}/\psi, \eta = \sum_{j \in Q} max(0, x_{i,j} - \delta_i),$
  $\phi = \sum_{j \in Q} max(0, \eta_j - x_{i,j})$
  if $\eta = 0$ i $\phi = 0$ **then**
    izađi iz algoritma
  end if
  for $j = 1$ do N **do**
    if $z_{i,j} = 1$ **then**
      if $x_{i,j} > \delta_j$ **then**
        $x_{i,j} = \delta_j$
      end if
      if $x_{i,j} < \varepsilon_j$ **then**
        $x_{i,j} = \varepsilon_j$
      end if
    end if
  end for
end while

Za ograničenje $\sum_{i=1}^{N} x_i = 1$, postavljeno je da je $\psi = \sum_{j \in Q} x_{i,j}$ i da je $x_{i,j} = x_{i,j}/\psi$ za sve akcije koje zadovoljavaju uslov da je $j \in Q$. Isti pristup za zadovoljavanje ograničenja je korišćen i u [152]. Da bi se osiguralo da je učešće svake akcije u portfoliju u okviru predefinisane donje i gornje granice ε i δ, respektivno, korišćeno je: $if\ \ x_{i,j} > \delta_{i,j}\ \ then\ \ x_{i,j} = \delta_{i,j}$ i $\ \ if\ \ x_{i,j} < \varepsilon_{i,j}\ \ then\ \ x_{i,j} = \varepsilon_{i,j}$.

Potrebno je napomenuti da u implementaciji mFA [147] nije korišćen pristup baziran na c-vrednostima za uključivanje i izbacivanje akcija iz portfolija kao u [152]. Na osnovu rezultata eksperimentata, zaključeno je da ovaj pristup ne poboljšava

performanse OPS algoritma, već samo povećava trošenje računarskih resursa.

U mFA metaheuristici funkcija podobnosti modelira privlačenje svitaca, gde je ova funkcija direktno proporcionalna sa privlačnošću. Vrednost funkcije podobnosti se računa za svakog svica nakon generisanja SN agenata prema istoj jednačini kao i u originalnom FA:

$$I(x) = \begin{cases} \frac{1}{1+f(x)}, \text{ ako je } f(x) > 0 \\ 1 + |f(x)|, \text{ u suprotnom,} \end{cases} \tag{5.39}$$

gde je $f(x)$ vrednost funkcije cilja.

Obzirom da je mFA adaptiran za rešavanje problema sa ograničenjima, u fazi inicijalizacije se takođe za svakog svica računa vrednost prekoračenja ograničenja CV prema jednačini:

$$CV_i = \sum_{g_j(x_i)>0} g_j(x_i) + \sum_{j=q+1}^{m} h_j(x_i), \tag{5.40}$$

gde su $g_j(x_i)$ ograničenja nejednakosti, a $h_j(x_i)$ ograničenja jednakosti. CV se kasnije koristi za proces selekcije pomoću Debovih pravila [10], [153].

Kretanje svitaca (osnovna jednačina pretrage) se modelira na isti način kao i u originalnoj FA implementaciji [54], prema jednačini (5.29).

Posle procesa pretrage izračunavanju se nove vrednosti varijabli odlučivanja [147]:

$$z_{i,k}^{t+1} = \text{round}(\frac{1}{1 + e^{-z_{i,k}^t + \phi_{i,j}(z_{i,k}^t - z_{j,k}^t)}}) - 0.06) \tag{5.41}$$

gde je $z_{i,k}^{t+1}$ varijabla odlučivanja za k-tu akciju novog rešenja, $z_{i,k}$ je varijabla odlučivanja k-te varijable starog rešenja, a $z_{j,k}$ definiše varijablju odlučivanja k-e akcije svetlijeg svica j. Primećuje se da se varijable odlučivanja generišu na malo drugačiji način nego u fazi inicijalizacije, gde se koristi (5.38).

Nakon što se generiše novo i-to rešenje pomoću jednačina (5.29) i (5.41), bolje rešenje između novog $x_i(t+1)$ i starog $x_i(t)$ se zadržava u procesu selekcije na

osnovu Debovih pravila.

Analizom originalnog FA uočen je nedostatak u snazi diversifikacije, koji je posebno naglašen u ranijim iteracijama izvršavanja algoritma. U ovoj fazi, balans između intenzifikacije i diversifikacije nije dobro uspostavljen, što se odražava na lošije najbolje i prosečne rezultate za probleme globalne optimizacije sa ograničenjima [147]. Zbog toga je hibridnu metaheuristiku mFA inkorporiran mehanizam pčele izviđača i parametar *limit* iz algoritma ABC.

Kada vrednost parametra *limit* za određenog agenta dostigne predefinisanu graničnu vrednost koja je nazvana prag granice (eng. limit threshold - LT), taj agent biva zamenjen slučajnim agentom koji se generiše pomoću jednačina (5.36) i (5.38). Vrednost ovog parametra je utvrđena putem serije empirijskih testova.

Međutim, u kasnijim iteracijama mFA hibrida, uz predpostavku da je algoritam pronašao domen prostora pretrage gde se nalazi optimum, intenzivna eksploracija više nije potrebna. Da bi se kontrolisalo izvršavanje pretrage pomoću eksploracije, uveden je novi kontrolni parametar koji je nazvan prelomna tačka eksploracije (eng. exploration breakpoint - EBP). Ovaj parametar kontroliše da li će diversifikacija da se izvrši ili neće da se izvrši u tekućoj iteraciji [147].

Tokom izvršavanja algoritma, α parametar se postepeno smanjuje sa svoje početne vrednosti prema jednačini:

$$\alpha(t) = (1 - (1 - ((10^{-4}/9)^{1/IN}))) * \alpha(t-1), \tag{5.42}$$

gde je t broj tekuće iteracije, a IN maksimalni broj iteracija.

Pseudo-kod mFA prikazan je u Algoritmu 13 [147]. Neki detalji su izostavljeni radi preglednosti. U prikazanom pseudo-kodu, SN je broj svitaca u populaciji, IN je ukupan broj iteracija algoritma, a t je trenutna iteracija.

Algoritam 13 Pseudo-kod mFA hibrida

generiši početnu populaciju svitaca x_i i z_i $(i = 1, 2, 3, ..., SN)$ pomoću jednačina (5.36) i (5.41)
primeni algoritam dogovora
jačina svetlosti I_i u tački x_i se definiše pomoću $f(x)$
definiši koeficijent apsorpcije svetlosti γ
definiši broj iteracija IN
izračunaj CV svih svitaca u populaciji korišćenjem jednačine (5.40)
postavi početnu vrednost parametra α
postavi $t = 0$
while $t < IN$ **do**
  for $i = 1$ do SN **do**
    for $j = 1$ do i **do**
      if $I_i < I_j$ **then**
        pomeri svica i prema svicu j u d dimenzija korišćenjem jednačina (5.29) i (5.41)
        privlačnost varira sa razdaljinom r pomoću $\exp[-\gamma r]$
        evauliraj novo rešenje, zameni staro sa boljim rešenjem korišćenjem Debovih pravila i ažuriraj intenzitet svetlosti
        if rešenje i nije poboljšano i $t \leq EBP$ **then**
          povećaj $limit_i$ za 1
        else
          postavi $limit_i$ na 0
        end if
      end if
    end for
  end for
  if $t > EBP$ **then**
    zameni sve agente za koje važi da je $limit > LT$ sa slučajnim agentima pomoću jednačina (5.36) i (5.41)
  end if
  primeni algoritam dogovora
  rangiraj sve svice i pronađi najboljeg
  izračunaj novu vrednost za parametar α pomoću (5.42)
end while

5.2.3.3 Eksperimentalni rezultati

Hibridna metaheuristika eFA testirana je na standardnim G funkcijama globalne

optimizacije sa ograničenjima [5], [12]. U Tabeli 5.5 prikazana je komparativna analiza rezultata FA i eFA metaheuristika.

Tabela 5.5: Komparativna analiza rezultata sa metaheuristikom FA

Problem	Optimum		FA	eFA
G1	-15.000	Najbolji	-12.432	**-15.000**
		Prosečni	-10.161	**-15.000**
		StDev	1.902	**0.000**
G2	-0.803619	Najbolji	-0.803619	-0.803619
		Prosečni	-0.782942	**-0.799479**
		StDev	0.022	**0.012**
G3	-1.000	Najbolji	-1.000	-1.000
		Prosečni	-1.000	-1.000
		StDev	0.000	0.000
G4	-30665.539	Najbolji	-30378.381	**30665.539**
		Prosečni	-30259.518	**30665.539**
		StDev	33.888	**0.000**
G5	5126.497	Najbolji	5126.497	5126.497
		Prosečni	**5126.835**	5128.135
		StDev	**0.065**	3.209
G6	-6961.814	Najbolji	-6961.811	**-6961.814**
		Prosečni	-6961.786	**-6961.814**
		StDev	0.011	**0.000**
G7	24.306	Najbolji	24.307	24.307
		Mean	24.311	24.311
		StDev	**0.012**	0.035
G8	-0.095825	Najbolji	-0.095825	-0.095825
		Prosečni	-0.095825	-0.095825
		StDev	0.000	0.000
G9	680.63	Najbolji	680.630	680.630
		Prosečni	680.630	680.630
		StDev	0.000	0.000
G10	7049.25	Najbolji	7072.411	**7049.331**
		Prosečni	7388.856	**7072.705**
		StDev	216.552	**25.302**
G11	0.75	Najbolji	0.750	0.750
		Prosečni	0.750	0.750
		StDev	0.000	0.000
G12	-1.000	Najbolji	-1.000	-1.000
		Prosečni	-1.000	-1.000
		StDev	0.000	0.000
G13	0.053950	Najbolji	0.054	0.054
		Prosečni	0.252	**0.219**
		StDev	0.102	**0.097**

Iz Tabele 5.5 vidi se da se osnovni FA zaglavljuje u lokalnom optimumu za mnoge test funkcije zato što nije dovoljno jak mehanizam eksploracije. Ovaj nedostatak najviše dolazi do izražaja u testovima *G1*, *G4*, *G6* i *G10*. eFA potpuno ispravlja ove nedostatke u slučajevima funkcija *G1*, *G4* i *G6* i postiže odlične rezultate i za najbolje i prosečne vrednosti i za standardnu devijaciju. U testu *G10* eFA postiže bolje rezultate od originalnog pristupa za sve indikatore performansi. eFA značajno poboljšava prosečne vrednosti u testu *G2*, za koji obe metaheuristike pronalaze optimalno rešenje. Slično važi i za *G13* benčmark.

S druge strane, hibridizacijom se generišu i neki gubici, ali uz odličan balans dobitaka i gubitaka. Prosečne vrednosti i standardna devijacija je nešto lošija u testu *G5*, dok rezultati optimizacije funkcije *G7* pokazuju malo lošiju standardnu devijaciju. U ova dva slučaja, eFA nije uspeo da pronađe pravi domen prostora pretrage u ranijim ciklusima u svim puštanjima algoritma.

Komparativna analiza rezultata eFA izvršena je i sa SAPF-GA [134], ABC [66], GI-ABC [130] i SOA-FS [146]. Rezultati ove analize prikazani su u Tabeli 5.6. Kao i u Tabeli 5.5, i u Tabeli 5.6, najbolji rezultati iz svake kategorije su prikazani masni slovima. Svi algoritmi su testirani sa 240,000 evaluacija funkcija, a prosečne vrednosti su računate na bazi 30 nezavisnih izvršavanja algoritama.

Na osnovu rezultata prikazanih u Tabeli 5.6 zaključuje se da je metaheuristika eFA u proseku bolja od svih drugih algoritama koji su uzeti za komparativnu analizu. Od navedenih algoritama, samo GI-ABC i SOA-FS pokazuju bolje rezultate od eFA za neke funkcije.

Tako na primer, u testu $G12$, samo eFA postiže optimalnu vrednost. Za isti test, bolje prosečne vrednosti generiše GI-ABC [130] zato što ova metaheuristika koristi izraženije ukrštanje najboljih rešenja u kasnijim ciklusima.

Kao još jedan reprezentativni primer, može da se navede $G5$ problem. U ovom slučaju, eFA postiže značajno bolje prosečne vrednosti i standardnu devijaciju od svih drugih metaheuristika. Ovo dalje znači da skoro u svakom izvršavanju eFA postiže optimalne rezultate.

Tabela 5.6: Komparativna analiza sa SAPF-GA, ABC, GI-ABC i SOA-FS

Problem	Opt.		SAPF-GA	ABC	GI-ABC	SOA-FS	eFA
G1	-15.000	Najbolji	-15.000	-15.000	-15.000	-15.000	-15.000
		Prosečni	-14.552	-15.000	-15.000	-15.000	-15.000
		StDev	N/A	0.000	0.000	0.000	0.000
G2	-0.803619	Najbolji	-0.803202	-0.803598	-0.803614	-0.803618	**-0.803619**
		Prosečni	-0.755798	-0.792412	**-0.800151**	-0.796049	-0.799479
		StDev	N/A	0.012	**0.007**	0.009	0.012
G3	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000	-1.000
		Mean	-0.964	-1.000	-1.000	-0.996	-1.000
		StDev	N/A	0.000	0.000	0.000	0.000
G4	-30665.539	Najbolji	-30665.401	-30665.539	-30665.539	-30665.539	-30665.539
		Prosečni	-30665.922	-30665.539	-30665.539	-30665.539	-30665.539
		StDev	N/A	0.000	0.000	0.000	0.000
G5	5126.497	Najbolji	5126.907	5126.484*	5126.497	5126.497	5126.497
		Prosečni	5214.232	5185.714	5164.762	5134.715	**5128.135**
		StDev	N/A	75.358	76.045	6.796	**3.209**
G6	-6961.814	Najbolji	-6961.046	-6961.814	-6961.814	-6961.814	-6961.814
		Prosečni	-6953.061	-6961.814	-6961.814	-6961.814	-6961.814
		StDev	N/A	0.000	0.000	0.000	0.000
G7	24.306	Najbolji	24.838	24.330	**24.306**	24.314	24.307
		Prosečni	27.328	24.473	24.395	24.331	**24.311**
		StDev	N/A	0.186	0.084	**0.015**	0.035
G8	-0.095825	Najbolji	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
		Prosečni	-0.095635	-0.095825	-0.095825	-0.095825	-0.095825
		StDev	N/A	0.000	0.000	0.000	0.000
G9	680.63	Najbolji	680.773	680.634	680.630	680.631	680.630
		Prosečni	681.246	680.640	680.635	680.634	**680.630**
		StDev	N/A	0.004	0.004	0.002	**0.000**
G10	7049.25	Najbolji	7069.981	7053.904	**7049.285**	7049.329	7049.331
		Prosečni	7238.964	7224.407	7192.397	7139.531	**7072.705**
		StDev	N/A	133.870	110.341	66.960	**25.302**
G11	0.75	Najbolji	0.750	0.750	0.750	0.750	0.750
		Prosečni	0.751	0.750	0.750	0.750	0.750
		StDev	N/A	0.000	0.000	0.000	0.000
G12	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000	-1.000
		Prosečni	-1.000	-1.000	-1.000	-1.000	-1.000
		StDev	N/A	0.000	0.000	0.000	0.000
G13	0.053950	Najbolji	0.054	0.760	0.054	0.054	0.054
		Prosečni	0.286	0.968	0.248	**0.138**	0.219
		StDev	N/A	**0.055**	0.155	0.096	0.097

Da bi performanse metaheuristike mFA mogle da se uporede sa drugim vodećim metaheuristikama prikazanim u objavljenim radovima, parametri su podešeni slično kao i u [152]. Parametar broja svitaca u populaciji SN se računa na sledeći način [147]:

$$SN = 20\sqrt{N}, \tag{5.43}$$

gde je N broj potencijalnih akcija u portfoliju.

Vrednost za maksimalni broj iteracija IN u jednom izvršavanju algoritma se određuje prema:

$$IN = \frac{1000 * N}{SN} \tag{5.44}$$

Kao što je već rečeno, da bi se pojačala moć diversifikacije originalnog FA, koristi se parametar *limit* koji je nasleđen iz metaheuristike ABC. *limit* je brojač koji prebrojava koliko puta je određeni agent (svitac) bezuspešno modifikovan. Kada vrednost *limit* dostigne predefinisani prag LT, određeni svitac se zamenjuje novim, slučajnim rešenjem. Vrednost LT zavisi od vrednosti za SN i IN [147]:

$$LT = \frac{IN}{SN} = \frac{\frac{1000*N}{SN}}{20\sqrt{N}} \tag{5.45}$$

Prelomna tačka eksploracije EBP kontroliše da li se koristi mehanizam diversifikacije. Prema izvedenim testovima [147], mFA generiše lošije rezultate ako se diversifikacija koristi u kasnijim ciklusima izvršavanja algoritma. U većini izvršavanja, algoritam je uspeo da pronađe pravi deo prostora pretrage tokom ranijih iteracija, pa se eksploracija u kasnijim iteracijama pokazala kao mehanizam koji smanjuje intenzifikaciju i bespotrebno troši resurse. Vrednost parametra EBP je empirijski utvrđena i postavljena na vrednost $IN/2$.

Početna vrednost FA parametra pretrage α je postavljena na vrednost od 0.5, ali se postepeno smanjuje prema jednačini (5.42).

Hibrid mFA za upravljanje ograničenjima koristi algoritam dogovora i granicu prekoračenja v sa dekrementom dec na isti način kao i hibridna metaheuristika GI-ABC. Početna vrednost parametra v je postavljena na 1.0, a dec na 1.001, dok je prag postavljen na vrednost od 0.0001. Ova problematika detaljno je opisana u Sekciji 5.1.3.

Za generisanje heurističke efikasne granice korišćeno je 51 različitih vrednosti λ, označene kao ξ, gde je ξ broj izvršavanja algoritma. Dakle, vrednost za $\Delta\lambda$ je postavljena 0.02 zato što λ u prvom puštanju algoritma ima vrednost 0, a u poslednjem 1.

Vrednost za broj akcija koje će biti uključene u portfolio K je postavljena na 10, donja granica težinskih koeficijenata akcija ε na 0.01, a gornja δ na 1. Budući da donja granica entropije zavisi od broja akcija u portfoliju, L_E je definisan u intervalu $[0, \ln 10]$.

U Algoritmu 14 prikazan je pseudo-kod mFA sa akcentom na podešavanjima parametara [147].

Da bi se parametri algoritma bolje diferencirali, oni su podeljeni u tri grupe: mFA globalni kontrolni parametri, FA parametri pretrage, parametri portfolio optimizacije i parametri za upravljanje ograničenjima [147]. Sve vrednosti parametara su sumirane u Tabeli 5.7.

Algoritam 14 Pseudo-kod mFA sa podešavanjima parametara

$\lambda = 0$
while $\lambda \leq 1$ **do**
  $SN = 20\sqrt{N}$
  postavi parametre portfolio problema K, ε i δ
  faza inicijalizacije
  algoritam dogovora
  računanje podobnosti
  postavi vrednost parametra *limit* na 0 i izralunaj LT prema jednačini (5.45)
  postavi početne vrednosti za υ i α
  $IN = \frac{1000N}{SN}$
  for t=1 do IN **do**
    kretanje svitaca
    primeni selekciju između starog i novog rešenja primenom Debovih pravila
    ako je neophodno, izvrši fazu diversifikacije
    algoritam dogovora
    rangiraj sve svice u populaciji i pronađi najboljeg
    ponovo izdračunaj vrednosti za υ i α
    $t++$
  end for
  λ=λ + $\Delta\lambda$
end while

U cilju testiranja performansi hibridne metaheuristike mFA izvršene su tri grupe eksperimenata [147]:

- Prvo je izvršena komparatvina analiza između mFA i originalnog FA za rešavanje problema optimizacije portfolija CCMV sa ograničenjem entropije. Na ovaj način najbolje mogu da se vide unapređenja koja generiše mFA u odnosu na osnovni FA;
- Takođe je prikazana komparavina analiza između mFA za rešavanje problema optimizacije portfolija CCMV sa i bez ograničenja entropije. U ovom testu je analiziran uticaj entropije na diversifikaciju portfolija;
- Konačno, u trećem testu, prikazana je komparativna analiza između mFA i drugih vrhunskih metaheuristika. Algoritam je upoređen sa PSO koga je

Tabela 5.7: Vrednosti parametara mFA algoritma

Parametar	**Vrednost**
mFA globalni kontrolni parametri	
Broj svitaca u populaciji (SN)	zavisi od N
Broj iteracija (IN)	zavisi od SN
Granica praga (LT)	zavisi od SN i IN
Prelomna tačka eksploracije (EBP)	zavisi od IN
FA parametri pretrage	
Početna vrednost parametra slučajnosti α	0.5
Privlačnost svitaca na razdaljini r=0, β_0	0.2
Koeficijent apsorpcije γ	1.0
Parametri optimizacije portfolija	
Broj potencijalnih akcija (N)	zavisi od problema
Broj akcija u portfoliju (K)	10
Početna vrednost parametra izbegavanja rizika (λ)	0
Broj različitih λ vredosti (ξ)	51
Donja granica težinskog koeficijenta akcije (ε)	0.01
Gornja granica težinskog koeficijenta akcije (δ)	1.0
Donja granica entropije (L_E)	$[0, \ln K]$
Parametri za upravljanje ograničenjima	
Početna vrednost tolerancije prekoračenja (υ)	1.0
Dekrement (dec)	1.002

prikazao Cura [152] i sa GA, TS i SA čiji su rezultati objavljeni u [154].

Svi rezultati dobijeni su testiranjima na opštim modelom efikasne granice koji obezbeđuje rešenje za problem formulisan jednačinama (5.30) - (5.33). Test podaci preuzeti su sa zvanične web adrese gde se nalaze primeri raznih test problema: http://people.brunel.ac.uk/ mastjjb/jeb/orlib/portinfo.html. Preuzeti benčmark podaci se odnose na nedeljne cene akcija za period mart 1992. godine - septembar 1997. godine sa sledećih berzi: kineske berze *Hang Seng* sa 31 akcijom, nemačke berze *Dax 100* sa 85 akcija, Engleske *FTSE 100* berze sa 89 akcija, berze *S&P 100* sa 98 akcija iz SAD-a, i japanse *Nikkei* berze sa 225 akcija.

Podaci su sačuvani u Excel spreadsheet-u, odakle se učitavaju u algoritam [147].

Korišćeni su sledeći test podaci: prosečan prinos, standardna devijacija prinosa za svaku akciju i korelacija između svakog mogućeg para akcija. Takođe, za potrebe generisanja standardne efikasne granice, korišćeni su prosečni prinos i varijansa prinosa za svaku akciju.

Pošto vrednosti parametara SN, MCN i LT zavise od dimenzje problema N (broj akcija u testu), u Tabeli 5.8, prikazane su konkretne vrednosti za sve pomenute berze. Realne vrednosti su zaokružene na najbližu celobrojnu vrednost.

Donja granica entropije za sve benčmark probleme je postavljena u opsegu 0 i $\ln 10$, zato što je vrednost K postavljena na 10 u svim test slučajevima. Testovi su vršeni na Intel CoreTM i7-4770K @4GHz procesoru sa 16GB RAM memorije, Windows 7 x64 Ultimate 64 operativnom sistemu i Visual Studio 2012 sa .NET 4.5 Framework okruženjem.

Nakon što mFA generiše skupove Pareto optimalnih portfolija, heuristička efikasna granica može da se prati. Upoređivana je standardna efikasna ganica za pet realnih benčmark problema sa heurističkom efikasnom granicom istog skupa podataka koju generiše mFA [147]. Za komparativnu analizu između standardne i heurističke efikasne granice, korišćeni su prosečna Euklidska udaljenost, varijansa greške prinosa i prosečna greška prinosa, kao i u [152]. Vreme izvršavanja mFA za svaki benčmark je takođe prikazano [147].

Za potrebe izračunavanja prosečne Euklidske udaljenosti, neka par $(v_i^s, r_i^s) : (i = 1, 2, 3, ..., 2000)$ označava varijansu i prosečni prinos tačke na standardnoj efikasnoj granici, i neka par $(v_j^h, r_j^h) = (i = 1, 2, 3, ..., \xi)$ označava varijansu i prosečni prinos tačke na heurističkoj efikasnoj granici. Onda se tačka na standardnoj efikasnoj granici koja je najbliža tački na heurističkoj efikasnoj granici, označena kao, $(v_{i,j}^s, r_{i,j}^s)$, računa pomoću Euklidske udaljenosti [147]:

$$i_j = \min_{i=1,2,...2000}(\sqrt{(v_i^s - v_j^h)^2 + (r_i^s - r_j^h)^2} \\ j = 1, 2, 3, ..., \xi \tag{5.46}$$

Na osnovu (5.46), prosečna Euklidska udaljenost se određuje kao:

Tabela 5.8: Specifični parametri za benčmark probleme [147]

Parametar	**Vrednost**
Hang Seng sa 31 akcijom	
Broj rešenja u populaciji (SN)	111
Broj ciklusa (IN)	279
Prag limita (LT)	3
Prelomna tačka eksploracije (EBP)	140
DAX 100 sa 85 akcija	
Broj rešenja u populaciji (SN)	185
Broj ciklusa (IN)	459
Prag limita (LT)	3
Prelomna tačka eksploracije (EBP)	230
FTSE 100 sa 89 akcija	
Broj rešenja u populaciji (SN)	189
Broj ciklusa (IN)	479
Prag limita (LT)	3
Prelomna tačka eksploracije (EBP)	240
S&P 100 sa 98 akcija	
Broj rešenja u populaciji (SN)	198
Broj ciklusa (IN)	494
Prag limita (LT)	3
Prelomna tačka eksploracije (EBP)	247
Nikkei sa 225 akcija	
Broj rešenja u populaciji (SN)	300
Broj ciklusa (IN)	750
Prag limita (LT)	3
Prelomna tačka eksploracije (EBP)	375

$$\frac{(\sum_{j=1}^{\xi}\sqrt{(v_{i,j}^{s}-v_{j}^{h})^{2}-(r_{i,j}^{s}-r_{j}^{h})^{2}})}{\xi} \tag{5.47}$$

Varijansa greške prinosa se izračunava pomoću sledeće jednačine:

$$(\sum_{j=1}^{\xi}100|v_{i,j}^{s}-v_{j}^{h}|/v_{j}^{h})\frac{1}{\xi} \tag{5.48}$$

Prosečna greška prinosa se računa prema:

$$(\sum_{j=1}^{\xi} 100|r_{i,j}^{s} - r_{j}^{h}|/r_{j}^{h})\frac{1}{\xi} \tag{5.49}$$

Rezultati prvog testa, gde je upoređivan mFA sa osnovnim FA za problem optimizacije portfolija CCMV sa ograničenjem entropije, prikazani su u Tabeli 5.9. Vremena izvršavanja se u ovom slučaju mogu porediti, pošto se oba algoritma izvršavaju na istoj računarskoj platformi. Radi lakšeg prikaza, bolji rezultati iz svih kategorija su označeni masnim slovima.

Tabela 5.9: Komparativna analiza mFA i FA za portfolio problem sa entropijom

Problem	**N**	**Indikatori performansi**	**FA**	**mFA**
Hang Seng	31	Prosečna Euklidska udaljenost	0.0006	**0.0003**
		Varijansa greške prinosa (%)	1.7092	**1.2387**
		Prosečna greška prinosa (%)	0.7172	**0.4715**
		Vreme izvršavanja	**18**	20
DAX 100	85	Prosečna Euklidska udaljenost	0.0032	**0.0009**
		Varijansa greške prinosa (%)	7.3892	**7.2569**
		Prosečna greška prinosa (%)	1.4052	**1.3786**
		Vreme izvršavanja	**67**	71
FTSE 100	89	Prosečna Euklidska udaljenost	0.0005	**0.0004**
		Varijansa greške prinosa (%)	**2.6391**	2.7085
		Prosečna greška prinosa (%)	**0.3025**	0.3121
		Vreme izvršavanja	**81**	94
S&P 100	98	P rosečna Euklidska udaljenost	0.0011	**0.0003**
		Varijansa greške prinosa (%)	3.9829	**3.6026**
		Prosečna greška prinosa(%)	1.0025	**0.8993**
		Vreme izvršavanja	**129**	148
Nikkei	225	Prosečna Euklidska udaljenost	0.0001	**0.0000**
		Varijansa greške prinosa (%)	1.7834	**1.2015**
		Prosečna greška prinosa (%)	0.7283	**0.4892**
		Vreme izvršavanja	**335**	367

Kao što može da se vidi iz Tabele 5.9, mFA postiže bolje rezultate u skoro svim testovima. Samo u slučaju greške varijanse prinosa i prosečne greške prinosa za

test $FTSE100$, originalni FA je uspeo da postigne bolje rezultate. U ovom testu, diversifikacija u ranim iteracijama nije potrebna pošto algoritam brzo konvergira ka optimalnom domenu prostora pretrage, i u slučaju mFA, agenti se bespotrebno troše na eksploraciju.

Sva tri indikatora - prosečna Euklidska udaljenost, varijansa greške prinosa i prosečna greška prinosa su značajnije bolji u mFA testovima za $HangSeng$, $DAX100$, $S\&P100$ i $Nikkei$ berze. Pošto mFA koristi diversifikaciju u ranijim ciklusima, vreme izračunavanja za sve testove je gore (veće) nego u slučajnu originalnog FA.

Iz ovog primera se vidi da se hibridizacijom ne može dobiti savršeni algoritam. I u slučaju hibridizacije važe pravila teoreme "nema besplatnog ručka". Ali, dobru hibridizaciju koja se kreira pažljivom analizom prednosti i nedostataka kombinovanih metaheuristika karakteriše povoljan balans dobitaka i gubitaka.

U drugom setu eksperimenata je vršena komparativna između mFA za problem optimizacije portfolija sa i bez ograničenja entropije, kako bi se video koji utcaj entropija ima na diversifikaciju portfolija. Problem optimizacije portfolija bez ograničenja entropije se modelira jednačinama (5.30 - 5.33).

Na osnovu eksperimentalnih rezultata prikazanih u Tabeli 5.10, jasno je da ograničenje entropije blago utiče na performanse portfolija. U optimizaciji portfolija sa ograničenjem entropije za testove *Hang Seng* i *S&P*, prosečna Euklidska udaljenost je malo bolja i zaključije se da je portfolio diversifikovaniji. I u drugim testovima, rezulati koji su generisani za ovaj indikator su slični. Takođe i u slučaju testova *Hang Seng*, *DAX 100*, *FTSE 100* i *S&P 100*, optimizacija portfolio modela sa entropijom postiže bolje vrednosti za varijansu greške prinosa i prosečnu grešku prinosa. Samo u slučaju Nikkei benčmarka, ovi indikatori imaju bolju vrednost kada se optimizuje model u kome nije uključena entropija. Pošto algoritam alocira dodatno vreme za optimizaciju modela sa entropijom, vreme izvršavanja je u ovom slučaju nešto gore osim za test Hang Seng, zato što ovaj problem koristi manji broj akcija od drugih benčmark problema.

Konačno, u trećem setu eksperimenata, prikazana je komparativna analiza između mFA za model bez ograničenja entropije i drugih vrhunskih metaheuristika. Imple-

Tabela 5.10: Komparativna analiza rezultata mFA za portfolio problem sa i bez entropije

Problem	N	Indikatori performansi	Bez entropije	Sa entropijom
Hang Seng	31	Prosečna Euklidska udaljenost	0.0004	**0.0003**
		Varijansa greške prinosa (%)	1.2452	**1.2387**
		Prosečna greška prinosa (%)	0.4897	**0.4715**
		Vreme izvršavanja	20	20
DAX 100	85	Prosečna Euklidska udaljenost	0.0009	0.0009
		Varijansa greške prinosa (%)	7.2708	**7.2569**
		Prosečna greška prinosa (%)	1.3801	**1.3786**
		Vreme izvršavanja	**70**	71
FTSE 100	89	Prosečna Euklidska udaljenost	0.0004	0.0004
		Varijansa greške prinosa (%)	2.7236	**2.7085**
		Prosečna greška prinosa (%)	0.3126	**0.3121**
		Vreme izvršavanja	**92**	94
S&P 100	98	Prosečna Euklidska udaljenost	0.0004	**0.0003**
		Varijansa greške prinosa (%)	3.6135	**3.6026**
		Prosečna greška prinosa (%)	0.8997	**0.8993**
		Vreme izvršavanja	**146**	148
Nikkei	225	Prosečna Euklidska udaljenost	0.0000	0.0000
		Varijansa greške prinosa (%)	**1.1927**	1.2015
		Prosečna greška prinosa (%)	**0.464**	0.4892
		Vreme izvršavanja	**360**	367

mentacija metaheuristika za problem optimizacije portfolija CCMV sa ograničenjem entropije nije pronađen u literaturi, pa je komparativna analiza sa drugim metaheuristikama mogla da se izvrši samo za model bez entropije.

Ovog puta je mFA upoređivan sa TS, GA, SA, čiji su rezultati preuzeti iz [154], i sa PSO koji je prikazan u [152]. Za sve algoritme korišćeni su isti test podaci i isti broj evaluacija funkcija. Kao i u prva dva eksperimenta, za indikatore performansi uzeti su prosečna Euklidska udaljenost, varijansa greške prinosa i prosečna greška prinosa.

Parametri mFA koji su prikazani u tabelama 5.7 i 5.8, uporedivi su sa parametrima

algoritama koji su uzeti za komparativnu analizu [152], [154]. Rezultati za prosečno vreme izvršavanja su neuporedivi, pošto je mFA hibrid testiran na drugačijoj računarskoj platformi. U eksperimentima prikazanim u radovima [154] i [152] korišćen je Pentium M 2.13 GHz sa 1 GB RAM memorije. U Tabeli 5.11, gde su prikazani rezultati komparativne analize, najbolji rezultati iz svake kategorije su označeni masnim slovima.

Potrebno je spomenuti da u literaturi postoje i druge implementacije metaheuristika rojeva za problem optimizacije portfolija CCMV, kao što su ABC [155] i hibridni ABC (eng. hybrid ABC - HABC) [156] koji imaju slične performanse.

Tabela 5.11: Komparativna analiza mFA sa drugim metaheuristikama

Problem	**N**	**Indikatori performansi**	**GA**	**TS**	**SA**	**PSO**	**mFA**
Hang Seng	31	Prosečna Euklidska udaljenost	0.0040	0.0040	0.0040	0.0049	**0.0003**
		Varijansa greške prinosa (%)	1.6441	1.6578	1.6628	2.2421	**1.2387**
		Prosečna greška prinosa (%)	0.6072	0.6107	0.6238	0.7427	**0.4715**
		Vreme izvršavanja	18	9	10	34	20
DAX 100	85	Prosečna Euklidska udaljenost	0.0076	0.0082	0.0078	0.0090	**0.0009**
		Varijansa greške prinosa (%)	7.2180	9.0309	8.5485	**6.8588**	7.2569
		Prosečna greška prinosa (%)	**1.2791**	1.9078	1.2817	1.5885	1.3786
		Vreme izvršavanja	99	42	52	179	71
FTSE 100	89	Prosečna Euklidska udaljenost	0.0020	0.0021	0.0021	0.0022	**0.0004**
		Varijansa greške prinosa (%)	2.8660	4.0123	3.8205	3.0596	**2.7085**
		Prosečna greška prinosa(%)	0.3277	0.3298	0.3304	0.3640	**0.3121**
		Vreme izvršavanja	106	42	55	190	94
S&P 100	98	Prosečna Euklidska udaljenost	0.0041	0.0041	0.0041	0.0052	**0.0003**
		Varijansa greške prinosa (%)	**3.4802**	5.7139	5.4247	3.9136	3.6026
		Prosečna greška prinosa(%)	1.2258	**0.7125**	0.8416	1.4040	0.8993
		Vreme izvršavanja	126	51	66	214	148
Nikkei	225	Prosečna Euklidska udaljenost	0.0093	0.0010	0.0010	0.0019	**0.0000**
		Varijansa greške prinosa(%)	1.2056	1.2431	1.2017	2.4274	**1.2015**
		Prosečna greška prinosa (%)	5.3266	0.4207	**0.4126**	0.7997	0.4892
		Vreme izvršavanja	742	234	286	919	367

Eksperimentalni rezultati prikazani u Tabeli 5.11 pokazuju da ni jedan od pet algoritama koji su korišćeni u komparativnoj analizu nemaju izražene prednosti, ali je u proseku mFA bolji pristup nego četiri druge metaheuristike.

mFA postiže bolje rezultate za prosečnu Euklidsku udaljenost u svih pet testova. U testovima *HangSeng* i $FTSE100$, mFA je bolji od svih drugih algoritama za sva

tri indikatora. U ovim testovima, mFA je uspeo da aproksimira standardnu efikasnu granicu sa manjom prosečnom greškom prinosa i varijansom greške prinosa sa istim nivoom rizika.

Drugi najbolji algoritam prikazan u Tabeli 5.11 je GA, koji postiže najbolji rezultat za prosečnu grešku prinosa i varijanse u testovima *DAX 100* i *S&P 100*, respektivno. TS pokazuje najbolje performanse za indikator prosečne greške prinosa za problem *S&P 100*, SA za prosečnu grešku prinosa u testu *Nikkei*, dok PSO pokazuje da je najrobusnija metaheuristika za indikator greške varijanse prinosa u testu *DAX 100*.

Na osnovu svih prikazanih testova i analiza, zaključuje se da mFA pokazuje najbolje performanse za problem optimizacije portfolija CCMV. Hibridizacijom sa metaheuristikom ABC ispravljen je problem nedostatka snage diversifikacije i uspostavljen je bolji balans između eksploatacije i eksploracije.

U ovom radu implementiran je još jedan hibrid za rešavanje problema optimizacije portfolija CCMV. Ovog puta je urađena hibridizacija algoritma ABC sa metaheuristikom FA.

Baš kao i mFA, ova hibridizacija spada u grupu hibrida niskog nivoa, gde je jednačina pretrage metaheuristike FA inkorporirana u algoritam ABC. Proces intenzifikacije osnovnog ABC nije dovoljno naglašen, što je u neposrednoj vezi sa lošim balansom između diversifikacije i intenzifikacije. S druge strane, metaheuristika FA koristi jak mehanizam intenzifikacije koji se sprovodi primenom operatora mutacije. Dakle, nedostatak ABC algoritma (slaba intenzifikacija) ispravljen je sa prednostima FA (jaka intenzifikacija). Hibridni pristup koji kombinuje ABC sa pretragom svitaca (eng. ABC firefly search - ABC-FS), u nastavku *ABC-FS* značajno popravlja proces intenzifikacije ABC algoritma i pa samim tim i brzinu konvergencije ka optimalnom domenu prostora pretrage [144]. S Obzirom da osnovni ABC koristi jak mehanizam diversifikacije baziran na pčelama izviđačima, ABC-FS poseduje sofisticiranu proceduru za izlazak iz lokalnog minimuma.

Osnovna jednačina pretrage metaheuristike FA enkapsulirana je u fazu pčele radilice ABC algoritma. U nekim iteracijama algoritam vrši standardnu ABC pretragu, dok u drugim koristi pretragu svitaca. Izbor pretrage se vrši probabilistički i ovaj pro-

ces kontroliše i usmerava dodatni kontroli parametar - okidač pretrage svitaca (eng. firefly search trigger - FST). Integrisanjem metode pretrage FA u ABC, poboljšanja je i snaga intenzifikacije i uspostavljen je bolji balans između intenzifikacije i diversifikacije, o čemu svedoče rezultati empirijskih testova.

ABC-FS je testiran na problemu optimizacije portfolija CCMV i pokazao se kao robusna i efikasna metaheuristička metoda [144]. Više o navedenm hibridu može se naći u radu [144].

5.2.4 Zaključak

Metaheuristika FA je noviji i vrlo uspešan algoritam testiran na velikom broju problema. Osnovna jednačina pretrage metaheuristike FA (5.29) koristi operator mutacije i za lokalnu i za globalnu pretragu. Primećeno je da ova jednačina ispoljava jaku intenzifikaciju, što za posledicu ima da algoritam brzo konvergira. Za uspostavljanje balansa između intenzifikacije i diversifikacije koristi se dinamički parametar α, čija se vrednost menja tokom izvršavanja algoritma. U ranijim fazama izvršavanja algoritma ovaj parametar ima veću vrednost, koja se postepeno smanjuje sa svakom iteracijom.

S obzirom da metaheuristika FA koristi privlačnost kao meru podobnosti rešenja, populacija se tokom procesa pretrage deli u grupe, gde svaka grupa vrši pretragu svog lokalnog domena. Zbog ovog svojstva metaheuristika FA na "prirodan" način rešava probleme sa više optimuma.

Iako se FA pokazao kao efikasan pristup za rešavanje mnogih benčmark i praktičnih problema, pažljivom analizom su uočeni neki nedostaci. Osnovni FA generiše dobre rezultate za probleme globalne optimizacije sa ograničenjima vrednosti parametara [54]. Međutim, kada se primeni na probleme sa ograničenjima, metaheuristika FA ne pokazuje zadovoljavajuće performanse. FA nema jasno naglašenu jednačinu diversifikacije i često se dešava da se algoritam zaglavi u suboptimalnim delovima prostora pretrage. Obzirom da ograničenja značajno sužavaju dopustivi domen, za rešavanje ovakvih problema je potrebna jača diversifikacija, što je potvrđeno i empirijskim testovima na standardnim benčmark problemima globalne optimizacije sa

ograničenjima.

Problem nedostatka snage diversifikacije implicira i problem neusklađenog balansa između eksploracije i eksploatacije, koji je "najopasniji" u ranijim ciklusima, kada algoritam još uvek nije pronašao optimalni domen prostora pretrage. U ranijim ciklusima se često dešava da se algoritam zaglavi u domenu lokalnog optimuma, što vodi lošijim najboljim i prosečnim rezultatima.

Problemi slabe diversifikacije i neodgovarajućeg balansa između eksploatacije i eksploracije osnovne metaheuristike FA za probleme sa ograničenjima mogu da se isprave hibridizacijom. Ova tvrdnja je dokazana na primerima dve hibridizacije sa ABC algoritmom.

Procedura pčele izviđača ABC algoritma, koja vrši proces eksploracije, se pokazala kao efikasna metoda za pronalaženje optimalnog dela prostora pretrage i za izlazak iz lokalnog optimuma. Integrisanjem ovog mehanizma u metaheuristiku FA jačina diversifikacije može značajno da se poboljša.

Prva hibridna implementacija - eFA - je prilagođena za rešavanje benčmark problema sa ograničenjima. Ova hibridna metaheuristika, inspirisana pčelom izviđačem ABC algoritma, uvodi novu vrstu veštačkog agenta u populaciju, tzv. "iscrpljenog" svica. Svici koji ne uspeju da poboljšaju rešenje koje eksploatišu u toku predefinisanog broja ciklusa postaju "iscrpljeni" i bivaju zamenjeni sa slučajnim rešenjem. Na ovaj način je u FA metaheuristiku inkorporiran i naglašen mehanizam diversifikacije.

Predpostvka metaheuristike eFA je da će nakon određenog broja ciklusa proces pretrage konvergirati ka optimalnom domenu i da tada više nema potrebe za diversifikacijom. U ovim fazama izvršavanja mehanizam diversifikacije se transformiše u proces intenzivne eksploatacije tako što se "iscrpljeni" svici zamenjuju hibridima koji se generišu ukrštanjem najboljih jedinki u populaciji.

Empirijskim testovima je utvrđeno da eFA ispravlja nedostatke osnovnog FA. Takođe, zbog pojačane intenzifikacije u kasnijim ciklusima, eFA generiše preciznija rešenja od mnogih drugih vrhunskih metaheuristika za globalne probleme sa ograničenjima.

Druga implementacija - mFA - spada u grupu hibrida niskog nivoa, gde je procedura pčele izviđača ugrađena u metaheuristiku FA [147]. mFA je prilagođen za rešavanje problema optimizacije portfolija sa ograničenjima kardinalnosti i entropije. Ovaj problem spada u grupu teških kombinovanih problema kvadratnog i celobrojnog programiranja.

Slično kao i u slučaju metaheuristike eFA, i mFA predpostavlja da je snažna eksploracija potrebna u ranijim ciklusima izvršavanja. U ovim ciklusima, eksploracija može da pomogne algoritmu da pronađe optimalni domen i da se izbavi iz lokalnih optimuma. Da bi validirao ovu predpostavku, u početnim fazama izvršavanja, mFA inkorporira mehanizam pčele izviđača. U kasnijim iteracijama se ovaj mehanizam prosto eliminiše i FA nastavlja svoju klasičnu pretragu. Isključivanje procedure diversifikacije se reguliše kontrolnim parametrom algoritma.

U cilju ispitivanja performansi metaheuristike mFA, izvršena su brojna empirijska testiranja i komparativne analize sa drugim vrhunskim algoritmima [147]. Empirijski testovi su pokazali da mFA značajno poboljšava balans između diversifikacije i intenzifikacije u ranijim iteracijama i da se više ne dešava da se algoritam zaglavjuje u suboptimalnim regionima. Komparativna analiza sa drugim metaheuristikama je pokazala da mFA ima superiorne performanse za problem optimizacije portfolija sa ograničenjima kardinalnosti i entropije.

Kao zaključak se navodi da je moguće ispraviti nedostatke osnovne metaheuristike FA pomoću hibridizacija. Hibridne implementacije ne samo da su poboljšale osnovni FA, već pokazuju i superiorne performanse u odnosu na druge vrhunske metaheuristike iz literature.

5.3 Hibridizacija metaheuristike SOA

5.3.1 Osnovna metaheuristika SOA

Proces pretage grupe ljudi inspirisao je razvoj metaheuristike SOA. Tvorci ovog algoritma su Dai, Chen i Zhu. Algoritam je prvo testiran na problemima globalne optimizacije bez ograničenja kompleksnih funkcija [55], [56]. Kasnije je prikazan još

jedan rad, gde je SOA primenjen na rešavanje globalne optimizacije i upoređivan sa DE algoritmom i tri modifikovane verzije metaheuristike PSO [63]. Napominje se da je SOA relativno skoro promenila ime na metaheuristiku optimizacije grupom ljudi (eng. human group optimizer - HGO) [57].

Proces ljudske pretrage koji simulira SOA zavisi od memorije, logičkog zaključivanja, prethodnog iskustva i interakcija između ljudi. SOA se razlikuje od drugih metaheuristika rojeva zato što se već na početku izvršavanja algoritma cela populacija deli na nekoliko jednakih (po broju jedinki) podpopulacija prema redosledu tragača. Individue koje pripadaju istoj podpopulaciji formiraju socijalnu jedinicu koja se naziva susedstvo, pri čemu svaka podpopulacija pretražuje samo svoj region prostora pretrage.

U osnovnoj implementaciji SOA populacija se deli na tri jednake podpopulacije. Populacija potencijalnih rešenja se predstavlja dvodimenzionom matricom, $P_{i,j}$ ($i = 1, 2, ..., SN,\ j = 1, 2, ..., D$), gde SN označava veličinu populacije, a D predstavlja veličinu problema. Svaki tragač i u populaciji predstavlja se vektorom $x_i(t) : (x_{i1}, x_{i2}, ..., x_{iD})$ u generaciji t. Obzirom da svaki tragač poseduje internu memoriju, tragač se definiše i sledećim atributima: vektor $x_i(t-1)$ čuva varijable problema iz prethodne generacije, vektor pb_i, koji se naziva lična najbolja pozicija, čuva najbolju poziciju svakog tragača od početka izvršavanja algoritma i vektor n_{best} koji označava najbolju istorijsku poziciju u susedstvu svakog tragača.

Algoritam sprovodi proces pretrage primenom sledećih komponenti [63]:

- proces ažuriranja rešenja sprovodi se pomoću smera pretrage i dužine koraka;
- smer pretrage se računa na osnovnu izbalansiranih vrednosti egoističkog, altruističkog i proaktivnog ponašanja tragača i
- dužina koraka se izračunava primenom metode fazi zaključivanja zbog činjenice da su ljudi ponekada kolebljivi prilikom donošenja odluka, i to može da se opiše jednostavnim „if-else" kontrolnim pravilima.

Smer pretrage računa se balansiranjem vrednosti egoističkog, altruističkog i proaktivnog ponašanja tragača. Svi tragači su pre svega egoisti, jer se ponašaju u skladu sa sopstvenim procenama, što se u algoritmu modelira tako što svaki tragač

prati svoju najbolju poziciju kada određuje smer pretrage. Egoistički smer vektora emulira ovo ponašanje [63]:

$$d_{i,ego}(t) = pb_i - x_i(t) \tag{5.50}$$

Tragač je takođe i socijalno biće i pokazuje karakteristike altruističkog ponašanja. U tom kontekstu tragač ima potrebu da komunicira sa drugim tragačima iz svog susedstva, da sarađuje sa njima i prikuplja informacije i da na osnovu toga donosi odluke. Altruistički smer pretrage svakog tragača i računa se kao vektor $d_{i,alt}$ pomoću jedne od sledećih jednačina:

$$d_{i,alt1} = n_{best} - x_i(t) \tag{5.51}$$

$$d_{i,alt2} = l_{best}(t) - x_i(t), \tag{5.52}$$

gde n_{best} predstavlja istorijsku najbolju poziciju u susedstvu, a l_{best} trenutno najbolje rešenje u susedstvu.

Konačno, svaki tragač ima karakteristike proaktivnog ponašanja, zato što se usmerava prema praktičnom cilju. Osim toga, buduće ponašanje može da se predvidi i usmerava se na osnovu ponašanja iz prošlosti. Proaktivni vektor smera prikazan je sledećom jednačinom:

$$d_{i,pro} = x_i(t_1) - x_i(t_2), \tag{5.53}$$

gde su t_1 i t_2 u intervalu $\{t, t-1, t-2\}$, dok $x_i(t_1)$ i $x_i(t_2)$ predstavljaju respektivno najbolje i najgore pozicije u skupu $\{x_i(t-2), x_i(t-1), x_i(t)\}$.

Konačno, smer tragača i se određuje stohastičkom kombinacijom egoističkog, altruističkog i proaktivnog vektora prema [63]:

$$d_t = \text{sgn}[\omega * d_{i,pro}(t) * \phi_1 * d_{i,ego}(t) * \phi_2 * d_{i,alt}(t)], \tag{5.54}$$

gde je ω kontrolni parametar, a ϕ_1 i ϕ_2 su pseudo-slučajni brojevi u intervalu $[0, 1]$. Parametar ω služi za postepeno smanjivanje intenziteta lokalne pretrage proaktivnog smera $d_{i,pro}$ i-tog tragača i za balansiranje globalne i lokalne pretrage. U većini implementacija, ovaj parametar se dinamički menja tako što se smanjuje od 0.9 do 0.1 tokom jednog izvršavanja algoritma. Funkcija sgn primenjuje se nad svakom varijablom ulaznog vektora.

Za izračunavanje dužine koraka koristi se fazi zaključivanje. Prvo se svaka populacija sortira u opadajućem redosledu na osnovu vrednosti funkcije cilja, pričemu svaka jedinka dobija redni broj, od 1 do SS, gde SS predstavlja veličinu podpopulacije kojoj tragač pripada. Redni brojevi se koriste kao ulaz u metodu fazi zaključivanja.

Stepen pripadnosti tragača i se računa prema sledećoj formuli:

$$u_i = u_{max} - \frac{SS - I_i}{SS - 1} * (u_{max} - u_{min}), \tag{5.55}$$

gde je I_i redni broj tragača x_i, u_{max} je najveći stepen pripadnosti i postavlja se obično na vrednost od oko 1.0. Sistem fazi zaključivanja koristi princip kontrolnog pravila oblika **if** $(uslov)$ **then** $(akcija)$. Belova funkcija pripadnosti koristi se kao akcija i računa se na sledeći način:

$$u(x) = e^{-x^2/2^{2*\delta}} \tag{5.56}$$

Zbog jednostavnosti se samo jedna varijabla uzima u obzir, a vrednosti stepena pripadnosti za ulazne varijable nalaze se u opsegu $[-3\delta, +3\delta]$, a pritom su manje od 0.0111. Dakle, vrednost u_{min} se postavlja na 0.0111. Parametar δ Belove funkcije pripadnosti računa se kao:

$$\delta = \omega * |X_{best} - X_{avg}|, \tag{5.57}$$

gde je X_{best} pozicija najboljeg tragača u podpopulaciji kojoj i-ti tragač pripada, a X_{avg} je prosečna pozicija svih tragača u istoj podpopulaciji. Vrednost δ je ista za sve agente u istoj podpopulaciji. Parametar ω se postepeno smanjuje tokom izvršavanja

algoritma, kako bi se povećala preciznost pretrage.

Da bi se poboljšala lokalna pretraga, vrednost u_i za i-tog tragača se računa posebno za svaku varijablu potencijalnog rešenja:

$$u_{i,j} = rnd(u_i, 1), \text{ za } j = 1, 2, ..., D, \tag{5.58}$$

gde je $rnd(u_i, 1)$ pseudo-slučajan broj u opsegu $[u_i, 1]$. Akcioni deo fazi zaključivanja, koji generiše dužinu koraka $a_{i,j}$ za j-tu varijablu i-tog tragača se računa prema:

$$a_{i,j} = \delta_j * \sqrt{-ln(u_{i,j})} \tag{5.59}$$

Konačno, ažuriranje pozicije varijable j svakog tragača i računa se pomoću vektora smera d i dužine koraka a [63]:

$$x_{i,j}(t+1) = x_{i,j}(t) + a_{i,j}(t) * d_{i,j}(t) \tag{5.60}$$

Jednačnina (5.60) predstavlja osnovnu jednačinu pretrage metaheuristike SOA.

Da bi se proces pretrage poboljšao, u svakoj generaciji (iteraciji), podpopulacije uče jedna od druge pomoću mehanizma koji se naziva učenje između podpopulacija (eng. inter-subpopulation learning). U originalnoj SAO implementaciji, svaka podpopulacija zamenjuje svoja dva najgora rešenja sa dve najbolje jedinke iz preostale dve podpopulacije.

Ako bi se proces pretrage usmeravao samo na osnovu lokalnih informacija svake podpopulacije, algoritam bi mogao da pokazuje nedostatak preuranjene konvergencije i da se zaglavi u prostoru lokalnih optimuma. Pomoću mehanizma učenja između podpopulacija, preuranjena konvergencija se izbegava i algoritam konvergira brže ka optimalnom rešenju.

SOA pseudo-kod je prikazan u Algoritmu 15. U pseudo-kodu, *nGen* je broj generacija u jednom izvršavanju algoritma.

Algoritam 15 Pseudo-kod metaheuristike SOA

generiši SN slučajnih tragača u populaciji
$t = 0$
while $t < nGen$ **do**
  for $i = 0$ to SN **do**
    izračunaj $d_i(t)$ pomoću jednačine (5.54)
    izračunaj $\alpha_i(t)$ pomoću jednačine (5.59)
    ažuriraj poziciju svakog tragača korišćenjem jednačine (5.60)
  end for
  evauliraj sve tragače i ažuriraj njihovu istorijsku najbolju poziciju
  pokreni mehanizam učenja između podpopulacija
  povećaj brojač trenutne generacija $(t + 1)$
end while
odredi najbolje rešenje u populaciji

5.3.2 Nedostaci metaheuristike SOA

Na osnovu podataka iz baze Scopus, može da se zaključi da je metaheuristika SOA uglavnom testirana na specifičnim problemima, za koje je algoritam adaptiran. Kao primer mogu da se navedu implementacije prikazane u [88] i [157]. U tim posebnim slučajevima SOA postiže zadovoljavajuće performanse, zato što su kontrolni parametri algoritma fino optimizovani prema specifičnim potrebama.

Međutim, u standardnim bazama (WoS, Scopus), nije pronađen rad koji testira ovaj algoritam na skupu opštih problema globalne optimizacije sa ograničenjima. Da bi se utvrdile performanse SOA za ovu klasu problema, sprovedeni su praktični eksperimenti [146]. Rezultati testiranja pružaju dobru osnovu za ispitivanje prednosti i nedostataka.

Pregled rezultata empirijskih testova osnovne verzije metaheuristike SOA na standardnim G funkcijama globalne optimizacije sa ograničenjima [12] dat je u Tabeli 5.12 [146]. Algoritam je testiran putem 30 nezavisnih izvršavanja i zabeležene su najbolje i srednje vrednosti i standardna devijacija.

Na osnovu prikazanih rezultata, zaključuje se da SOA pokazuje slabosti za ovakvu vrstu problema. Prvo, proces eksploatacije koji se usmerava metodom fazi zaključi-

vanja nije dovoljno intenzivan u slučaju mnogih benčmark problema da bi algoritam konvergirao ka optimalnom regionu prostora pretrage. Čak i za relativno lak optimizacioni problem, kao što je funkcija $G1$, ovaj algoritam nije uspeo da pronađe optimum, a prosečne vrednosti su značajno lošije od rezultata drugih metaheuristika, čiji su rezultati prikazani u objavljenim radovima.

Osim navedenog, zapaženo je da se mehanizam učenja između podpopulacija poziva previše često, što za posledicu ima preuranjenu konvergenciju podpopulacija koja algoritmu još više otežava da se približi optimumu. Zamena potpunih rešenja između podpopulacija daje rezultate u ranijim iteracijama, ali bi u kasnijim ciklusima proces pretrage trebalo preusmeriti na fino istraživanje u okolini najboljih rešenja [146].

Daljom analizom osnovnog SOA, utvrđeno je da samo u nekoliko izvršavanja algoritam ima sreće i uspeva da pogodi pravi deo prostora pretrage i za neke funkcije pronalazi optimalno (ili blizu optimalnog) rešenje. Međutim, u ostalim izvršavanjima, algoritam promašuje region u kome se nalazi najbolje rešenje, što kao posledicu ima lošije srednje vrednosti i standardnu devijaciju.

U testu $G2$, najbolje rešenje koje je pronašao originalni SAO daleko je od optimalnog, a prosečne vrednosti i standardna devijacija su loše. U ovom slučaju, algoritam ni u jednom izvršavanju nije uspeo da pronađe optimalni region. Slično važi i za $G7$ test za koji SOA pokazuje loše performanse i za najbolje i za prosečne vrednosti.

S druge strane, u optimizaciji problema $G8$ i $G9$, SOA je pronašao optimum, ali su prosečne vrednosti slabije. U nekim izvršavanjima, kada algoritam pronađe pravi region, mehanizmom učenja između podpopulacija pretraga konvergira do optimuma. Međutim, u ostalih izvršavanjima, pravi region se ne pronalazi, što vodi ka lošijih prosečnim vrednostima.

U $G10$ testu postignute su loše performanse za sva tri indikatora - najbolje i prosečno rešenje i standardna devijacija. Kada je u pitanju optimizacija funkcija $G5$ i $G13$, SOA je konvergirao ka optimumu, ali bi prosečni rezultati i standardna devijacija trebalo da budu bolji. Konačno, rezultati za $G3$, $G4$, $G6$, $G11$ i $G12$ funkcije su zadovoljavajući.

Tabela 5.12: Rezultati osnovnog SOA za globalne probleme sa ograničenjima

Problem	Optimum		SOA
G1	-15.000	Najbolji	-14.758
		Prosečni	-13.536
		StDev	0.383
G2	-0.803619	Najbolji	-0.785725
		Prosečni	-0.717389
		StDev	0.061
G3	-1.000	Najbolji	-1.000
		Prosečni	-0.994
		StDev	0.000
G4	-30665.539	Najbolji	-30665.539
		Prosečni	-30665.539
		StDev	0.000
G5	5126.497	Najbolji	5126.497
		Prosečni	5164.766
		StDev	36.78
G6	-6961.814	Najbolji	-6961.814
		Prosečni	-6961.814
		StDev	0.000
G7	24.306	Najbolji	24.385
		Prosečni	24.472
		StDev	0.103
G8	-0.095825	Najbolji	-0.095825
		Prosečni	-0.085924
		StDev	0.008
G9	680.63	Najbolji	680.632
		Prosečni	680.649
		StDev	0.006
G10	7049.25	Najbolji	7128.405
		Prosečni	7435.702
		StDev	330.25
G11	0.75	Najbolji	0.750
		Prosečni	0.750
		StDev	0.000
G12	-1.000	Najbolji	-1.000
		Prosečni	-1.000
		StDev	0.000
G13	0.053950	Najbolji	0.054
		Prosečni	0.314
		StDev	0.283

5.3.3 Hibridizacija SOA sa metaheuristikom FA

Obzirom da metaheuristika FA koristi sofisticirani mehanizam intenzifikacije, nedostatak SOA algoritma mogao bi da se ispravi hibridizacijom sa ovim algoritmom. Navedeno potvrđuju i empirijski testovi na standardnim G funkcijama za globalnu

optimizaciju sa ograničenjima.

Na osnovu rezultata u Tabeli 5.4 [146], gde su prikazane performanse metaheuristike FA, i Tabeli 5.12 [146], gde su navedeni rezultati SOA algoritma, zaključuje se da FA postiže mnogo bolje rezultate za benčmark probleme $G2$, $G7$ i $G10$, i to za sva tri indikatora. U testovima $G2$, $G3$, $G5$, $G9$ i $G13$, FA postiže mnogo bolje prosečne vrednosti i standardnu devijaciju. S druge strane, u slučaju funkcija $G1$ i $G4$, SOA nadjačava FA, ali uz napomenu da oba algoritma pokazuju lošije performanse u testu $G1$. Za ostale benčmark probleme, FA i SOA imaju slične karakteristike.

Posmatrano u proseku, performanse i jednog i drugog algoritma slabije su za probleme globalne optimizacije sa ograničenjima. Međutim, ova dva pristupa su komplementarna. Problemi za koje SOA generiše lošije rezultate, FA pokazuje dobre performanse, i obrnuto.

U cilju unapređenja metaheuristike SOA, urađena je hibridizacija sa FA [146]. Hibridizacija je inspirisana idejom da se proces pretrage grupe ljudi olakša dodatnim mehanizmima. Tako na primer, ako ljudi pretragu izvode u mraku, mogli bi da koriste baklje, kako bi bolje videli teren koji pretražuju, i da poput svitaca, budu privučeni od strane drugih ljudi čije baklje imaju jači intenzitet svetlosti.

Hibridna metaheuristika niskog nivoa, SOA - pretraga svitaca (eng. SOA firefly search - SOA-FS), enkapsulira jednačinu pretrage FA u proces ažuriranja individua u populaciji metaheuristike SOA [146]. Takođe, SOA-FS koristi modifikovani mehanizam učenja između podpopulaciji koji rešava problem preuranjene konvergencije između podpopulacija.

U fazi inicijalizacije, algoritam generiše slučajnu populaciju od SN individua (tragača):

$$x_{i,j} = lb_j + \text{rand}(0,1) * (ub_j - lb_j), \tag{5.61}$$

gde je $x_{i,j}$ j-ti parametar i-og tragača, $rand(0,1)$ je pseudo-slučajan broj uniformno distribuiran između 0 i 1, a ub_j i lb_j su gornja i donja granica j-og parametra potencijalnog rešenja, respektivno.

Nakon faze inicijalizacije, populacija se deli na tri jednake podpopulacije prema rednim brojevima tragača, kao i u osnovnom SOA algoritmu. Podobnost, koja zavisi od vrednosti funkcije cilja, u slučaju problema minimizacije, računa se kao:

$$fitness_i = \begin{cases} 1/(1+f_i), & \text{ako je } f_i \geq 0 \\ 1+|f_i|, & \text{u suprotnom} \end{cases} \tag{5.62}$$

gde je f_i vrednost funkcije cilja koja se optimizuje.

U fazi inicijalizacije za svakog tragača u populaciji računa se vrednost prekoračenja dopustivog prostora CV prema sledećoj jednačini:

$$CV_i = \sum_{j:\ g_j(x_i)>0} g_j(x_i) + \sum_{j=q+1}^{m} |h_j(x_i)| \tag{5.63}$$

gde su $g_j(x_i)$ ograničenja nejednakosti, a $h_j(x_i)$ ograničenja jednakosti. CV_i se kasnije koristi za proces selekcije pomoću Debovih pravila [10], [153].

U SOA-FS implementaciji [146], proces ažuriranja rešenja u populaciji je modifikovan hibridizacijom sa pretragom svitaca, tako što je u osnovni algoritam ugrađena još jedna jednačina pretrage [54]:

$$x_i(t) = x_i(t) + \beta_0 * e^{-\gamma * r_{i,k}^2}(x_k - x_i) + \alpha * (\theta - 0.5) \tag{5.64}$$

gde je β_0 privlačenje svitaca na razdaljini od r=0, γ označava varijabilnost privlačnosti, α je parametar slučajnosti, θ označava pseudo-slučajan broj između 0 i 1, i $r_{i,k}$ je razdaljina između tagača i i tragača k. Privlačnost je direktno proporcionalna sa funkcijom podobnosti.

Udaljenost između tragača i i k se računa prema sledećoj jednačini:

$$r_{i,k} = ||x_i - x_k|| = \sqrt{\sum_{j=1}^{D}(x_{i,j} - x_{k,j})^2}, \tag{5.65}$$

gde je D broj varijabli problema koji se optimzuje.

U većini slučajeva, $\beta_0 = 0$ i $\alpha \in [0, 1]$ vrednosti daju zadovoljavajuće rezultate, ali su u slučaju SOA-FS hibrida empirijski utvrđene [146]. Parametar α je dinamički parametar koji se menja tokom jednog izvršavanja algoritma. Parametar γ spada u grupu važnijih parametara jer on utiče na brzinu konvergencije procesa pretrage.

SOA-FS koristi dodatne parametre koji kontrolišu proces pretrage. Prvi predloženi parametar je ponašanje pretrage (eng. search behavior - SB), koji se podešava u opsegu $[0, 1]$. Ovaj parametar određuje koja će jednačina pretrage da se koristi u tekućoj iteraciji i radi na sledeći način: ako je $\zeta \leq SB$, pretraga se izvodi pomoću FA ažuriranja, a u suprotnom slučaju koristi se SOA mehanizam intenzifikacije. ζ je pseudo-slučajan broj uniformno generisan između 0 i 1 i računa se posebno za svaku generaciju. Na osnovu navedenog, jasno je da je osnovni SOA specijalni slučaj SOA-FS, kada se vrednost SB postavi na 0.

Nakon što se generiše novo rešenje, bilo pomoću osnovne SOA jednačine (5.60), bilo FA pretragom (5.64), pobednik između novog ($x_{i,new}$) i starog ($x_{i,old}$) rešenja se zadržava u procesu selekcije primenom Debovih pravila, i prenosi se u sledeću generaciju.

SOA-FS takođe uvodi izvesne modifikacije u mehanizam učenja između podpopulacija [146]. Kao najvažnije, smanjena je učestalost pozivanja ovog mehanizma. Kao što je već i navedeno, analizom originalnog SOA, zaključeno je da, ako se mehanizam učenja poziva previše često, da podpopulacije prebrzo konvergiraju jedna ka drugoj i da algoritam može da se zaglavi u nekom od suboptimalnih domena. Zbog toga, SOA-FS ne poziva mehanizam učenja u svakoj generaciji.

Osim frekventnosti pozivanja, predložen je i novi mehanizam učenja, kao i parametar koji ga kontroliše. Drugi predloženi parametar nazvan je pozivanje mehanizma učenja (eng. learning mechanism invocation - LMI) koji proces pretrage kontroliše tako što se u prvih LMI generacija koristi isti mehanizam učenja koji se primenjuje i u originalnom SOA, a to je da se pozicije dve najgore individue iz svake podpopulacije razmenjuju sa najboljim individuama iz preostale dve podpopulacije. U kontekstu SOA-FS za ovaj stari mehanizam predložen je naziv SOA učenje. Međutim, nakon LMI generacija, novi mehanizam učenja, koji je nazvan SOA-FS mehanizam, se ko-

risti. Novi mehanizam poboljšava intenzifikaciju tako što se najgore rešenje iz svake podpopulacije zamenjuje sa rešenjem mutantom. Mutant se generiše ukrštanjem najboljih rešenja iz sve tri podpopulacije [146]:

$$\begin{aligned} SP_{1,worst,j} &= SP_{1,best,j}, \text{ ako je } \sigma \in [0, 0.33) \\ SP_{1,worst,j} &= SP_{2,best,j}, \text{ ako je } \sigma \in [0.33, 0.66) \\ SP_{1,worst,j} &= SP_{3,best,j}, \text{ ako je } \sigma \in [0.66, 1], \end{aligned} \tag{5.66}$$

gd je $SP_{1,worst,j}$ j-ti parametar najgoreg rešenja u prvoj podpopulaciji, $SP_{1,best,j}$, $SP_{2,best,j}$ i $SP_{3,best,j}$ predstavljaju j-te parametre najboljeg rešenja u prvoj, drugoj i trećoj podpopulaciji, respektivno, a σ je pseudo-slučajan broj uniformno distribuiran između 0 i 1.

Dakle, u kasnijim iteracijama podpopulacije uče jedna od druge tako što međusobno razmenjuju cela rešenja. Na ovaj način se održava snaga diversifikacije u granicama jedne podpopulacije, tako što svaka podpopulacija istražuje regione drugih podpopulacija. Međutim, u kasnijim generacijama, uz predpostavku da je algoritam pronašao optimalni domen, podpopulacije uče jedna od druge ukrštanjem najboljih rešenja. Ovim mehanizmom pojačava se intenzifikacija oko najbolje pronađenih rešenja. Korišćenjem opisanog mehanizma učenja, postiže se bolji balans između eksploatacije i eksploracije, a samim tim i algoritam postiže bolje rezultate.

Metaheuristika SOA-FS koristi isti pristup za upravljanje ograničenjima jednakosti kao i hibridni pristup GI-ABC [130] (jednačina 5.9). U ovom slučaju, jako je važno da se tolerancija prekoračenja ε postavi na odgovarajuću vrednost pošto kvalitet rezultata u velikoj meri od nje zavisi.

Vrednost tolerancije prekoračenja se postepeno smanjuje tokom izvršavanja algoritma prema [132]:

$$\varepsilon(t+1) = \frac{\varepsilon(t)}{dec} \tag{5.67}$$

Vrednost dec je postavljena na 1.001, a prag za ε na 0.0001.

Pseudo-kod SOA-FS je dat u Algoritmu 16 [146]. Neki implementacioni detalji su izostavljeni radi preglednosti.

Algoritam 16 Pseudo-kod hibridne metaheuristike SOA-FS

```
inicijalizuj SN slučajnih tragača u populaciji
podeli populaciju na tri podpopulacije
izračunaj odgovarajuće vrednosti za sve tragače korišćenjem jednačina (5.62) i (5.63)
sortiraj sva rešenja u populaciji na osnovu vrednsti funkcije podobnosti
postvi početne vrednosti za ω, α i ε
t=0
while t < nGen do
  if ζ ≤ SB then
    izvrši ažuriranje rešenja pomoću SOA pretrage
  end if
  if ζ > SB then
    izvrši ažuriranje rešenja pomoću FA pretrage
  end if
  izaberi i zadrži bolje rešenje primenom Debovih pravila
  evauliraj sve tragače u populaciji i ažuriraj najbolje rešenje
  if t % 5 == 0 then
    pozovi mehanizam učenja između podpopulacija
    if t ≤ LMI then
      primeni SOA mehanizam učenja
    else
      primeni SOA-FS mehanizam učenja
    end if
  end if
  evauliraj sve tragače u populaciji i ažuriraj najbolje rešenje
  sortiraj sve tragače prema podbnosti kao priprema za ažuriranje u sledećoj generaciji
  zapamti najbolje rešenje u svakoj podpopulaciji
  povećaj brojač trenutne generacije (t + 1)
  ponovo izračunaj vrednosti za ω, α i ε
  zapamti najbolje rešenje za sve podpopulacije
end while
```

5.3.3.1 Eksperimentalni rezultati

Da bi se ispitale performanse hibridne metaheuristike SOA-FS [146], izvršena su

testiranja na 13 standardnih G benčmark funkcija za globalnu optimizaciju sa ograničenjima [5], [12]. Za svaku test instancu, algoritam je nezavisno izvršavan 30 puta, i za svako izvršavanje koristilo se drugačije seme pseudo-slučajnog broja. Kao indikatori performansi, uzeto je najbolje i prosečno rešenje i varijansa.

Osnovni parametri algoritma podešeni su na način da se tokom jednog izvršavanja algoritma vrši 240,000 evaluacija funkcija, kako bi performanse algoritma bile uporedive sa drugim pristupima [66, 130]. Broj tragača u populaciji SN postavljen je na 80, a broj generacija u jednom izvršavanju algoritma $nGen$ na 3000. Parametar koji određuje ponašanje procesa pretrage SB je postavljen na vrednost 0.5, dok je ispitivanjima algoritma određeno da se optimalna vrednost parametra LMI izračunava prema izrazu $0.75 * nGen$, što u ovom konkretnom slučaju iznosi 2250 [146]. SOA parametar ω se postepeno smanjuje tokom izvršavanja algoritma sa svoje početne vrednosti na vrednost od 0.1. Parametar α FA pretrage se takođe postepeno smanjuje prema sledećoj jednačini:

$$\alpha(t) = (1 - (1 - ((10^{-4}/9)^{1/nGen}))) * \alpha(t-1) \tag{5.68}$$

Konačno, treći po redu dinamički parametar ε računa se prema jednačini (5.67).

Obzirom da SOA-FS koristi parametre različitih algoritama, radi bolje preglednosti svi parametri su podeljeni u tri grupe: SOA-FS globalni parametri (skup parametara koji određuje opšte ponašanje algoritma), SOA parametri pretrage (skup parametara koji služi za podešavanje pretrage osnovne metaheuristike SOA), FA parametri pretrage (skup parametara koji služe za upravljanje FA procesom pretrage) i parametri za upravljanje ograničenjima. Ova kategorizacija prikazana je u Tabeli 5.13.

Iako neki od navedenih parametara imaju poznate i preporučene vrednosti u literaturi, izvršena su dodatna testiranja, kako bi se utvrdile optimalne vrednosti parametara u slučaju metaheuristike SOA-FS [146]. U Tabeli 5.14 dat je uprošćen primer takvih testiranja. Za parametre koji su specifični za FA algoritam, za potrebe testiranja, izabrane su sledeće vrednosti: $\alpha = 0.5$, $\beta_0 = 0.2$ i $\gamma = 1.0$. SOA-FS je testiran sa različitim vrednostima parametra α (od 0.1 do 1.0) sa inkrementom od

Tabela 5.13: Parametri metaheuristike SOA-FS

Parametar	**Vrednost**
Globalni parametri SOA-FS	
Broj tragača (SN)	80
Broj generacija ($nGen$)	3000
Ponašanje pretrage (SB)	0.5
Pozivanje mehanizma učenja (LMI)	2250
Parametri pretrage SOA	
Početna vrednost inercione težine (ω)	0.9
Maksimalni stepen pripadnosti (u_{max})	1.0
Minimalni stepen pripadnosti (u_{min})	0.0111
Parametri pretrage FA	
Početna vrednost parametra slučajnosti (α)	0.5
Privlačnost na udaljenosti r=0 (β_0)	0.2
Koeficijent apsorpcije (γ)	1.0
Parametri za upravljanje ograničenjima	
Početna vrednost tolerancije prekoračenja (ε)	1.0
Dekrement (dec)	1.002

0.1, bez menjanja druga dva parametra (β_0 i γ) [146]. Slična testiranja su izvršena i sa parametrom β_0 (od 0.1 do 1.0) istim inkrementom i γ (od 0.1 do 1.9) sa inkrementom od 0.3. Tabela 5.14 prikazuje komparativnu analizu rezultata najbližih suseda predloženom izboru parametara. Ako je izabrani set parametra ostvario bolje rezultate, u tabeli je prikazan znak plus, a ako je susedni set parametra ostvario bolje rezultate, u tabeli je stavljen znak minus, i konačno, ako su rezultati isti, polje u tabeli je prazno. Ako određeni skup parametara ima mnogo znakova plus u tabeli, to je indikacija da je dati set parametara dobar.

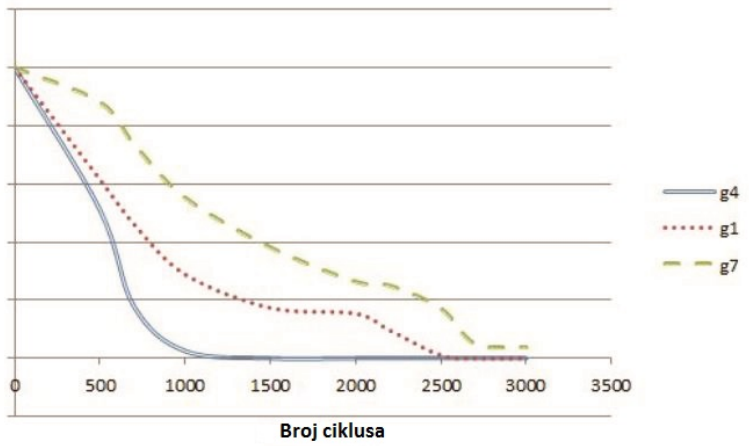
Radi detaljnije analize metaheuristike SOA-FS, istražena je i brzina konvergencije. Utvrđeno je da je 240,000 evaluacija dobar izbor, pošto je toliki broj evaluacija dovoljan da bi svaki problem konvergirao ka optimumu.

Tabela 5.14: Poređenje SOA-FS sa različitim podešavanjima parametara

Problem		α		β_0		γ	
		0.4	0.6	0.1	0.3	0.7	1.3
G1	Najbolji						
	Prosečni						
	StDev						
G2	Najbolji	+		+		+	+
	Prosečni	+	-	+	-	+	+
	StDev	+		+	-	+	+
G3	Najbolji						+
	Prosečni			+	+	+	+
	StDev						+
G4	Najbolji						
	Prosečni						
	StDev						
G5	Najbolji	+	+		+		
	Prosečni	+	+	-	+	-	-
	StDev	+	+	-	+	-	-
G6	Najbolji						
	Prosečni						
	StDev						
G7	Najbolji	+	+	+	+	+	
	Prosečni	-	+	+	+	+	+
	StDev	-	+	+	+	+	+
G8	Najbolji						
	Prosečni						
	StDev						
G9	Najbolji					+	+
	Prosečni	+				+	+
	StDev		+		+		
G10	Najbolji	+		+		+	+
	Prosečni	+	-	+	-	-	-
	StDev	+	-	+	-	-	-
G11	Najbolji						
	Prosečni						
	StDev						
G12	Najbolji						
	Prosečni						
	StDev						
G13	Najbolji		+		+	+	+
	Prosečni	-	+	-	+	+	+
	StDev	-	+	-	+	+	+

Neke funkcije, koje SOA-FS može da optimizuje relativno lako, već u ranijim iteracijama konvergiraju ka optimumu, a u nekim slučajevima i za manje od 1,000 iteracija. Takav slučaj se vidi kod optimizacije funkcija $G4$, $G8$, $G11$ i $G12$. Funkcije $G1$, $G7$ i $G10$ dostižu optimum (ili stagniraju) za manje od 3,000 ciklusa, dok je funkcijama $G2$, $G3$, $G5$, $G9$ i $G13$ potrebno 3,000 ciklusa da bi se stabilizovale.

SOA-FS ne može da dostigne optimum za $G7$ i $G10$ probleme, ali u ovim slučajevima algoritam postiže najbolje rešenje već posle 2,700 ciklusa i onda se zaglavi. U testovima $G2$ i $G9$, algoritam konvergira sporo ka najboljem rešenju koje može da postigne. Grafikoni konvergencije ovih karakterističnih funkcija su prikazani na Slici 5.2. Problem $G4$ konvergira brzo, funkciji $G1$ je potrebno skoro 3,000 iteracija, a funkcija $G7$ sporo napreduje i dostiže optimum. Radi lakše komparacije, vrednosti na grafiku su skalirane.



Slika 5.2: Grafik konvergencije za benčmark funkcije $G1$, $G4$ i $G7$

U Tabeli 5.15, prikazana je komparativna anliza između SOA-FS i originalnog SOA. Na osnovu rezultata iz tabele, vidi se da metaheuristika SOA-FS koriguje nedostatke osnovnog SOA.

Najznačajnija razlika u rezultatima između SOA-FS i SOA može da se vidi u test problemima $G1$, $G2$ i $G10$. U $G1$ testu, SOA-FS je ispravio loše prosečne vrednosti orginalnog SOA, koje imaju istu vrednost kao i najbolji rezultati, dok je u $G2$ testu hibrid postigao odlične rezultate, za razliku od osnovnog SOA. Za test $G10$, SOA-FS postiže odlične prosečne vrednosti. U $G13$ i $G5$ problemima, oba algoritma su konvergirala ka optimumu, ali su prosečne vrednosti i standardna devijacija značaj-

nije bolje u slučaju metaheuristike SOA-FS. Slična situacija je i u slučaju $G9$ testa. Kada je u pitanju $G7$ optimizacija, SOA-FS je u značajnoj meri prevazišao osnovni SOA u svim indikatorima performansi, dok je u $G8$ testu SOA-FS ispravio prosečne vrednosti osnovnog algoritma. Konačno, u $G3$, $G4$, $G6$, $G11$ i $G12$ testovima, oba algoritma su pokazala slične performanse.

Tabela 5.15: Upoređenje SOA i SOA-FS

Prob.	Opt.		SOA	SOA-FS
G1	-15.000	Najbolji	-14.758	**-15.000**
		Prosečni	-13.536	**-15.000**
		StDev	0.383	**0.000**
G2	-0.803619	Najbolji	-0.785725	**-0.803618**
		Prosečni	-0.717389	**-0.796049**
		StDev	0.061	**0.009**
G3	-1.000	Najbolji	-1.000	-1.000
		Prosečni	-0.994	**-0.996**
		StDev	0.000	0.000
G4	-30665.539	Najbolji	-30665.539	-30665.539
		Prosečni	-30665.539	-30665.539
		StDev	0.000	0.000
G5	5126.497	Najbolji	5126.497	5126.497
		Prosečni	5164.766	**5134.715**
		StDev	36.78	**6.796**
G6	-6961.814	Najbolji	-6961.814	-6961.814
		Prosečni	-6961.814	-6961.814
		StDev	0.000	0.000
G7	24.306	Najbolji	24.385	**24.314**
		Prosečni	24.472	**24.331**
		StDev	0.103	**0.015**
G8	-0.095825	Najbolji	-0.095825	-0.095825
		Prosečni	-0.085924	**-0.095825**
		StDev	0.008	**0.000**
G9	680.63	Najbolji	680.632	**680.631**
		Prosečni	680.649	**680.634**
		StDev	0.006	**0.002**
G10	7049.25	Najbolji	7128.405	**7049.329**
		Prosečni	7435.702	**7139.531**
		StDev	330.25	**66.96**
G11	0.75	Najbolji	0.750	0.750
		Prosečni	0.750	0.750
		StDev	0.000	0.000
G12	-1.000	Najbolji	-1.000	-1.000
		Prosečni	-1.000	-1.000
		StDev	0.000	0.000
G13	0.053950	Najbolji	0.054	0.054
		Prosečni	0.314	**0.138**
		StDev	0.283	**0.096**

Komparativna analiza između SOA-FS i originalnog FA prikazana je u Tabeli 5.16. Iz tabele je jasno da je SOA-FS bolja metaheuristika.

Tabela 5.16: Upoređenje FA i SOA-FS

Prob.	Opt.		FA	SOA-FS
G1	-15.000	Najbolji	-12.432	**-15.000**
		Prosečni	-10.161	**-15.000**
		StDev	1.902	**0.000**
G2	-0.803619	Najbolji	**-0.803619**	-0.803618
		Prosečni	-0.782942	**-0.796049**
		StDev	0.022	**0.009**
G3	-1.000	Najbolji	-1.000	-1.000
		Prosečni	-1.000	-1.000
		StDev	0.000	0.000
G4	-30665.539	Najbolji	-30378.381	**30665.539**
		Prosečni	-30259.518	**30665.539**
		StDev	33.888	**0.000**
G5	5126.497	Najbolji	5126.497	5126.497
		Prosečni	**5126.835**	5134.715
		StDev	**0.065**	6.796
G6	-6961.814	Najbolji	-6961.811	**-6961.814**
		Prosečni	-6961.786	**-6961.814**
		StDev	0.011	**0.000**
G7	24.306	Najbolji	**24.307**	24.314
		Prosečni	**24.311**	24.331
		StDev	**0.012**	0.015
G8	-0.095825	Najbolji	-0.095825	-0.095825
		Prosečni	-0.095825	-0.095825
		StDev	0.000	0.000
G9	680.63	Najbolji	**680.630**	680.631
		Prosečni	**680.630**	680.634
		StDev	**0.000**	0.002
G10	7049.25	Najbolji	7072.411	**7049.329**
		Prosečni	7388.856	**7139.531**
		StDev	216.552	**66.96**
G11	0.75	Najbolji	0.750	0.750
		Prosečni	0.750	0.750
		StDev	0.000	0.000
G12	-1.000	Najbolji	-1.000	-1.000
		Prosečni	-1.000	-1.000
		StDev	0.000	0.000
G13	0.053950	Najbolji	0.054	0.054
		Prosečni	0.252	**0.138**
		StDev	0.102	**0.096**

SOA-FS postiže značajno bolje rezultate od FA u testovima $G1$, $G4$, $G6$ i $G10$ za sva tri indikatora performansi. U $G13$ testu, SOA-FS postiže mnogo bolje rezultate za prosečne vrednosti i standardnu devijaciju. U benčmark problemu $G2$, SOA-FS

uspostavlja mnogo bolji balans između najboljih i prosečnih vrednosti, tako što se uz blago pogoršavanje najboljih rešenja, značajno poboljšavaju prosečni rezultati i standardna devijacija. S druge strane, FA postiže malo bolje rezultate od SOA-FS u testovima $G7$ i $G9$ za sva tri indikatora, a za prosečne vrednosti i standardnu devijaciju u testu $G5$.

Zbog potpunijeg uvida u performanse hibridne metaheuristike, predložena je i komparativna analiza između SOA-FS i druge tri metaheuristike: SAPF-GA (self-adaptive genetic algorithm) koga su implementirali Tassema i Yen [134], ABC algoritmom za globalnu optimizaciju sa ograničenjima koga su prikazali Karaboga i Akay [66] i GI-ABC koji je takođe deo ovog istraživanja [130]. Rezultati komparativne analize prikazani su u Tabeli 5.17.

Na osnovu rezultata eksperimenata prikazanih u Tabeli 5.17, zaključuje se da SOA-FS pokazuje bolje performanse od GI-ABC algoritma [130]. Kada se posmatraju najbolji rezultati, SOA-FS generiše kvalitetnija rešenja od GI-ABC u testovima $G7, G9$ i $G10$, dok GI-ABC postiže bolje rezultate za standardnu devijaciju u testu $G2$. Ako se posmatraju prosečne vrednosti i standardna devijacija, SOA-FS je bolji od GI-ABC u benčmark problemima $G5, G7, G9$ i $G10$. U benčmarku $G13$, oba algoritma postižu optimalne rezultate, dok je SOA-FS značajno bolji u standardnoj devijaciji. Kod drugih test problema, oba aloritma postižu slične performanse. Kao zaključak, naovodi se da je GI-ABC, kao vrhunska metaheuristika bolja od SOA-FS u postizanju optimalnih rezultata za neke funkcije, dok je s druge strane SOA-FS mnogo robusniji algoritam.

U komparaciji sa metaheuristikom ABC (Karaboga i Akay, 2011) [66], SOA-FS pokazuje značajno bolje rezultate u benčmark problemima $G2, G5, G7, G9, G10$ i $G13$. Rezultat funkcije $G5$ (označen sa * u Tabeli 5.17) koji je prijavljen u [66] bolji je od poznatog optimuma, što je posledica narušavanja ograničenja dopustivog prostora. ABC je bolji od SOA-FS samo u rezultatima za $G13$ za standardnu devijaciju. Za sve ostale test probleme, oba pristupa pokazuju slične rezultate.

SAPF-GA pokazuje slabe performanse u odnosu na druge algoritme. SOA-FS je bolji od SAPF-GA za skoro sve benčmark probleme, u svim indikatorima. Samo u

Tabela 5.17: Komparativna analiza sa SAPF-GA, ABC, GI-ABC i SOA-FS

Problem	Opt.		SAPF-GA	ABC	GI-ABC	SOA-FS
G1	-15.000	Najbolji	-15.000	-15.000	-15.000	-15.000
		Prosečni	-14.552	-15.000	-15.000	-15.000
		StDev	N/A	0.000	0.000	0.000
G2	-0.803619	Najbolji	-0.803202	-0.803598	-0.803614	**-0.803618**
		Prosečni	-0.755798	-0.792412	**-0.800151**	-0.796049
		StDev	N/A	0.012	**0.007**	0.009
G3	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000
		Prosečni	-0.964	-1.000	-1.000	-0.996
		StDev	N/A	0.000	0.000	0.000
G4	-30665.539	Najbolji	-30665.401	-30665.539	-30665.539	-30665.539
		Prosečni	-30665.922	-30665.539	-30665.539	-30665.539
		StDev	N/A	0.000	0.000	0.000
G5	5126.497	Najbolji	5126.907	5126.484*	5126.497	5126.497
		Prosečni	5214.232	5185.714	5164.762	**5134.715**
		StDev	N/A	75.358	76.045	**6.796**
G6	-6961.814	Najbolji	-6961.046	-6961.814	-6961.814	-6961.814
		Prosečni	-6953.061	-6961.814	-6961.814	-6961.814
		StDev	N/A	0.000	0.000	0.000
G7	24.306	Najbolji	24.838	24.330	**24.306**	24.314
		Prosečni	27.328	24.473	24.395	**24.331**
		StDev	N/A	0.186	0.084	**0.015**
G8	-0.095825	Najbolji	-0.095825	-0.095825	-0.095825	-0.095825
		Prosečni	-0.095635	-0.095825	-0.095825	-0.095825
		StDev	N/A	0.000	0.000	0.000
G9	680.63	Najbolji	680.773	680.634	**680.630**	680.631
		Prosečni	681.246	680.640	680.635	**680.634**
		StDev	N/A	0.004	0.004	**0.002**
G10	7049.25	Najbolji	7069.981	7053.904	**7049.285**	7049.329
		Prosečni	7238.964	7224.407	7192.397	**7139.531**
		StDev	N/A	133.870	110.341	**66.96**
G11	0.75	Najbolji	0.750	0.750	0.750	0.750
		Prosečni	0.751	0.750	0.750	0.750
		StDev	N/A	0.000	0.000	0.000
G12	-1.000	Najbolji	-1.000	-1.000	-1.000	-1.000
		Prosečni	-1.000	-1.000	-1.000	-1.000
		StDev	N/A	0.000	0.000	0.000
G13	0.053950	Najbolji	0.054	0.760	0.054	0.054
		Prosečni	0.286	0.968	0.248	**0.138**
		StDev	N/A	**0.055**	0.155	0.096

testu $G12$, oba algoritma pokazuju slične rezultate.

Na osnovu komparativne analize prikazane u Tabeli 5.17, zaključuje se da je u proseku SOA-FS mnogo bolja metaheuristika od drugih algoritama koji su uključeni u komparativnu analizu [146].

Treba da se kaže i to da je hibridizovani GI-ABC upoređivan sa drugim algoritmima

[130], od kojih je bolji, tako da je SOA-FS bolja i od sledećih algoritama: SF-ABC Mezure-Montesa i drugih [158] i od ATMES koga su prikazali Wang i drugi [159]. Osim ovoga, ABC (Karaboga i Akay, 2011) [66] je upoređivan sa drugih 9 algoritama, pa je SOA-FS bolji i od njih: HM [9], SR citenew23, ISR [160], OPA citenew24, ASCHEA [161], GA [162], SMES [162], PSO [163] i DE. Većina navedenih algoritama koriste veći broj evaluacija funkcija (350,000).

5.3.4 Zaključak

SOA simulira proces pretrage tima ljudi i pokazao se kao dobra metaheuristička metoda za praktične probleme. SOA pokazuje zadovoljavajuće rezultate na problemima bez ograničenja, ali efikasnost algoritma nije dovoljno validirana na skupu opštih problema globalne optimizacije sa ograničenjima i zbog toga je, između ostalog, predložena i implementacija SOA za ovu klasu problema [146].

Testiranja na standardnim G funkcijama globalne optimizacije sa ograničenjima [12] pokazala su da SOA ne postiže zadovoljavajuće rezultate [146]. Utvrđeno je da samo u nekoliko izvršavanja algoritam konvergira ka optimumu i za neke funkcije pronalazi optimalno (ili bluzu optimalnog) rešenje. Međutim, u ostalim izvršavanjima SOA ne konvergira dovoljno brzo, što za posledicu ima slabije prosečne vrednosti i standardnu devijaciju.

Ispitivanjem algoritma utvrđeno je da SOA nema dovoljno naglašen mehanizam intenzifikacije koji se usmerava samo metodom fazi zaključivanja. Čak i za relativno lak optimizacioni problem, kao što je funkcija $G1$, ovaj algoritam ne uspeva da pronađe optimum, a prosečne vrednosti su značajno lošije od rezultata drugih metaheuristika.

Kao drugo, navodi se da proces diversifikacije koji se izvodi zamenom potpunih rešenja između podpopulacija dalje rezultate u ranijim iteracijama, ali u kasnijim ciklusima ovaj mehanizam donosi više štete nego koristi, zato što bi tada pretragu trebalo preusmeriti na fino istraživanje u okolini najboljih rešenja [146]. Takođe, zapaženo je da ovakav mehanizam ponekada dovodi do problema preuranjene konvergencije između podpopulacija.

Pošto je utvrđeno da metaheuristika FA ima napredni mehanizam intenzifikacije, prošlo se od predpostavke da se navedeni nedostaci SOA mogu ispraviti hibridizacijom sa ovim pristupom. Navedeno potvrđuju i empirijski testovi na standardnim G funkcijama, gde SOA i FA generišu komplementarne rezultate.

Predloženi hibrid SOA-FS spada u grupu hibrida niskog nivoa jer je osnovna jednačina metaheuristike FA enkapsulirana u metaheuristiku SOA [146]. Pomoću dodatnog kontrolnog parametra SB određuje se da li će se proces ažuriranja rešenja u tekućoj iteraciji vršiti pomoću SOA ili FA jednačine pretrage. Takođe, da bi se sprečila preuranjena konvergencija, i da bi algoritam u kasnijim iteracijama izvršavao finu pretragu oko najboljih rešenja, SOA-FS koristi dinamičku proceduru učenja između podpopulacija čiji način izvršavanja zavisi od trenutne iteracije.

U početnim iteracijama, SOA-FS koristi isti mehanizam učenja između podpopulacija kao i osnovna metaheuristika SOA, tako što se dve najgore individue iz svake podpopulacije razmenjuju sa najboljim individuama iz preostale dve podpopulacije. U kasnijim ciklusima koristi se novi mehanizam učenja koji najgore rešenje iz svake podpopulacije zamenjuje rešenjem mutantom koje se generiše uniformnim ukrštanjem najboljih individua iz sve tri podpopulacije.

Empirijskim testovima na standardnim funkcijama sa ograničenjima potvrđeno je da SOA-FS ispravlja sve nedostatke osnovnog pristupa. Algoritam je pronašao optimume za probleme koje originalni algoritam nije uspeo da optimizuje, a prosečne vrednosti i standardna devijacija su značajno poboljšane [146]. Za svaki test problem, algoritam je uspeo da pronađe optimalni region prostora pretrage, a novi mehanizam učenja između podpopulacija omogućio je finu pretragu oko najboljih rešenja i predupredio je problem preuranjene konvergencije.

U komparaciji sa drugim metaheuristikama, čiji su rezultati prikazani u objavljenim radovima, SOA-FS postiže superiornije performanse.

5.4 Hibridizacija metaheuristike FWA

5.4.1 Osnovna metaheuristika FWA

Algoritam vatrometa (eng. fireworks algorithm - FWA) inspirisan je procesom eksplozije vatrometne rakete. FWA je relativno nova metaheuristika koju su kreirali Tan i Zhu 2010. godine za globalnu optimizaciju bez ograničenja [59]. Proces eksplozije vatrometne rakete, kada čitav spektar varnica obasipa nebo oko tačke gde je raketa lansirana, može da se posmatra i u kontekstu procesa pretrage.

FWA se bazira na predpostavci da postoje vatrometne rakete boljeg i slabijeg kvaliteta. U slučaju eksplozije vatrometne rakete visokog kvaliteta, varnice se grupišu u centru eksplozije (intenzifikacija) [59]. S druge strane, kod vatrometnih raketa slabijeg kvaliteta, samo mali broj varnica je grupisan u centru eksplozije, dok je većina varnica razbacana u prostoru (diversifikacija).

FWA emulira ova dva ponašanja vatrometnih raketa. U prvom slučaju, kada je vrednost funkcije cilja vatrometne rakete (potencijalnog rešenja problema) blizu optimuma, onda se takva raketa smatra dobrom, a to se modelira tako što eksplozija ima manju amplitudu, a pritom generiše veći broj varnica. U suprotnom slučaju, kada je vrednost funkcije cilja nekog rešenja daleko od optimuma, onda se proglašava da je ta vatrometna raketa loša, a to se modelira tako što eksplozija ima veliku amplitudu i generiše mali broj varnica. Prvi slučaj modelira proces intenzifikacije, a drugi diversifikacije.

U svakoj generaciji eksplozija bira se n lokacija za postavljanje n vatrometnih raketa. Nakon toga, na osnovu trenutnih lokacija varnica i vatrometnih raketa, bira se n drugih lokacija za sledeću iteraciju izvršavanja algoritma.

Osnovni parametri metaheuristike FWA su broj varnica i amplituda eksplozije. Osnovni algoritam razvijen je za probleme globalne optimizacije bez ograničenja:

$$\min \ f(x),\ x = (x_1, x_2, x_3, ..., x_D) \in S, \tag{5.69}$$

gde je x realni vektor sa $D \geq 1$ komponentom (varijablom), a $S \in R^D$ predstavlja hiper-pravougani prostor pretrage sa D dimenzija koje su omeđane donjim i gornjim granicama:

$$lb_i \leq x_i \leq ub_i, \quad i \in [1, D], \tag{5.70}$$

gde su lb_i i ub_i donja i gornja granica i-te varijable, respektivno.

Na osnovu ovakve formulacije problema, broj varnica koji se generiše eksplozijom vatrometne rakete x_i može da se definiše kao [59]:

$$s_i = m \frac{y_{max} - f(x_i) + \eta}{\sum_{i=1}^{n}(y_{max} - f(x_i)) + \eta}, \tag{5.71}$$

gde je m parametar koji kontroliše ukupan broj varnica koje se generišu eksplozijom n vatrometnih raketa, $y_{max} = \max(f(x_i)),\ i = 1, 2, ..., n$ je najgora vatrometna raketa u populaciji, tj. vatrometna raketa koja ima najveću vrednost funkcije cilja u slučaju problema minimizacije, dok η predstavlja malu konstantu koja služi za izbegavanje greške deljenja sa nulom.

Empirijskim testovima se došlo do zaključka da algoritam ne postiže dobre rezultate ako se s_i postavi na veliku vrednost. Iz tog razloga neophodno je da se odrede donja i gornja granica parametra s_i prema sledećoj formuli [59]:

$$\hat{s_i} = \begin{cases} \text{round}(\alpha \cdot m), & \text{ako je } s_i < \alpha \cdot m \\ \text{round}(\beta \cdot m), & \text{ako je } s_i > \beta \cdot m \ i \ \alpha < \beta < 1, \\ \text{round}(s_i), & \text{u suprotnom,} \end{cases} \tag{5.72}$$

gde su parametri α i β konstante.

Drugi važan parametar metaheuristike FWA je amplituda eksplozije. Kod visokokvalitetnih vatrometnih raketa, ova amplituda je manja nego kod manje kvalitetnih, što može da se modelira sledećom jednačinom:

$$A_i = \hat{A} \cdot \frac{f(x_i) - y_{min} + \eta}{\sum_{i=1}^{n}(f(x_i) - y_{min}) + \eta}, \tag{5.73}$$

gde $\hat{A}$ predstavlja najveću vrednost amplitude eksplozije, dok $y_{min} = \min(f(x_i))$, $i = 1, 2, ..., n$ označava najbolju vatrometnu raketu u populaciji od n vatrometnih raketa.

Proces eksplozije utiče na z slučajnih dimenzija (smerova) obuhvaćenih varnica. Broj dimenzija na koje eksplozija utiče se određuje jednačinom:

$$z = \text{round}(d \cdot \chi), \tag{5.74}$$

gde d označava broj varijabli problema koji se optimizuje (x lokacija), dok je χ psudo-slučajan broj uniformno distribuiran između 0 i 1.

Da bi se utvrdile lokacije varnica vatrometne rakete x_i, prvo mora da se utvrdi lokacija varnice $\hat{x}_j$. Ovaj proces je prikazan pseudo-kodom u Algoritmu 17 [59]. U priloženom pseudo-kodu, σ je pseudo-slučajan broj iz intervala $[-1, 1]$.

Algoritam 17 Pseudo-kod FWA algoritma za utvrđivanje lokacije varnice $\hat{x}_j$

```
utvrđivanje početne lokacije varnica x̂_j = x_i
izaberi slučajnih z dimenzija x̂_j pomoću jednačine (5.74)
izračunati promenu lokacije: h = A_i · σ
for svaku izabranu dimenziju x̂_k^j od x̂_j do
    x̂_k^j = x̂_k^j + h
    if x̂_k^j < x_k^min ili x̂_k^j > x_k^max then
        mapiraj x̂_k^j na prostor potencijalnih rešenja
    end if
end for
```

Za održavanje diversifikacije varnica (rešenja u populaciji) koristi se Gausov metod. Funkcija $Gaussian(1, 1)$, koja modelira Gausovu distribuciju prosečne vrednosti 1 i standardne devijacije 1, koristi se da bi se odredili koeficijenti pravaca eksplozije. Pomoću ovog metoda, u svakoj iteraciji se generiše $\hat{m}$ Gausovih varnica.

Pseudo-kod u Algoritmu 18 objašnjava proces inicijalizacije Gausovih varnica [59].

Algoritam 18 Pseudo-kod FWA algoritma za inicijalizaciju Gausovih varnica

utvrđivanje početne lokacije varnica: $\hat{x}_j = x_i$
izaberi slučajnih z dimenzija $\hat{x}_j$ pomoću jednačine (5.74)
izračunavanje koeficijenata Gausove eksplozije
$g = \text{Gaussian}(1, 1)$
for svaku izabranu dimenziju $\hat{x}_k^j$ od $\hat{x}_j$ **do**
  $\hat{x}_k^j = \hat{x}_k^j \cdot g$
  if $\hat{x}_k^j < x_k^{min}$ ili $\hat{x}_k^j > x_k^{max}$ **then**
    mapiraj $\hat{x}_k^j$ na prostor potencijalnih rešenja
  end if
end for

U svakoj iteraciji bira se n lokacija vatrometnih raketa. Trenutna najbolja lokacija x^* se uvek čuva i kao takva se prenosi u sledeću iteraciju. Dakle, ova metaheuristika koristi neki vid elitističke strategije. Nakon ovoga bira se *n-1* lokacija vatrometnih raketa na osnovu njihove udaljenosti od drugih lokacija. Razdaljina između lokacije x_i i drugih lokacija izračunava se pomoću:

$$R(x_i) = \sum_{j \in K} d(x_i, x_j) = \sum_{j \in K} ||x_i - x_j||, \tag{5.75}$$

gde je K skup svih trenutnih lokacija vatrometnih raketa i varnica.

Verovatnoća da će lokacija x_i biti izabrana računa se pomoću [59]:

$$p(x_i) = \frac{R(x_i)}{\sum\limits_{j \in K} R(x_j)} \tag{5.76}$$

Konačno, ceo proces izvršavanja algoritma može da se prikaže pseudo-kodom u Algoritmu 19.

U svakoj iteraciji generišu se dva tipa varnica prema Algoritmu 17 i Algoritmu 18. Broj varnica prvog tipa i amplituda eksplozije zavise od kvaliteta vatrometne rakete $f(x_i)$. Međutim, broj varnica druge vrste, koje se kreiraju procesom Gausove eksplozije, je nepromenljiv i kontroliše se parametrom algortima $\hat{m}$.

Algoritam 19 Pseudo-kod FWA algoritma

izaberi n slučajnih lokacija vatrometnih raketa
while se ne postigne maksimalni broj evaluacija funkcija **do**
  postavi n vatrometnih raketa na n lokacija
  for svaku vatrometnu raketu x_i **do**
    izračunaj broj varnica za vatrometnu raketu $\hat{s}_i$ korišćenjem jednačine (5.72)
    odredi lokacije $\hat{s}_i$ varnica vatrometne rakete x_i pomoću pseudo-koda 17
  end for
  for $k = 1 : \hat{m}$ **do**
    izaberi slučajanu vatrometnu raketu x_j
    generiši Gausovu varnicu za izabranu vatrometnu raketu korišćenjem pseudo-koda 18
  end for
  izaberi najbolju vatrometnu raketu za sledeću iteraciju
  izaberi slučajnih *n-1* lokacija iz skupa lokacija dve vrste varnica i trenutnih pozicija vatrometnih raketa sa verovatnoćom koja se računa pomoću jednačine (5.76)
end while

Kada se odrede lokacije obe vrste varnica, n lokacija se biraju za sledeću iteraciju. Na taj način, FWA izvodi približno $n + n + \hat{m}$ evaluacija funkcija u svakoj iteraciji [59].

Pregledom literature ustanovljeno je da, u odnosu na druge algoritme rojeva, postoji mali broj implementacija ovog algoritma. Međutim, iako malobrojne, implementacije potvrđuju da FWA mnogo obećava, kako u domenu globalne optimizacije bez ograničenja, tako i u domenu globalne optimizacije sa ograničenjima.

5.4.2 Nedostaci metaheuristike FWA

Analizom osnovne metaheuristike FWA utvrđeno je da ne postoji eksplicitna kontrola nad balansom između intenzifikacije i diversifikacije, niti je jasno kada i u kojoj meri metaheuristika FWA izvodi intenzifikaciju, a kada diversifikaciju.

Broj varijabli optimizacionog problema koje će biti izložene promenama u procesu pretrage određuje se jednačinom (5.74). Ova jednačina određuje z slučajnih dimenzija varnica (potencijalnih rešenja problema) koje će biti izložene promenama. Od vrednosti pseudo-slučajnog broja χ u jednačini (5.74) zavisi da li će pretraga biti više

lokalno, ili globalno orjentisana. Ako je vrednost ovog parametra manja, pretraga se usmerava lokalno, tj. vrši se proces intenzifikacije, a u drugom slučaju izvodi se diversifikacija, tj. globalna pretraga.

U početku izvršavanja algoritma diversifikacija bi trebala da bude izraženija, a u kasnijim ciklusima, intenzifikacija, kako bi metaheuristika izvodila finu pretragu oko optimuma. Međutim, u FWA pristupu se tokom celog izvršavanja algoritma procesi intenzifikacije i diversifikacije izvršavaju ravnomerno na osnovu vrednosti parametra χ.

Ispitivanjem optimizacije globalnih problema sa ograničenjima varijabli, utvrđeno je da algoritam konvergira previše sporo ka optimumu i da je potrebno da se snaga intenzifikacije pojača. Dalje je zaključeno da se za ovakve probleme algoritam retko zaglavljuje u suboptimalnim domenima, pa jaka diversifikacija koja se izvodi pomoću Gausovih varnica nije potrebna (Algoritam 18). Takođe je uočeno i da proces inicijalizacije Guasovih varnica značajno povećava troškove izračunavanja po iteraciji.

Predpostavlja se da u slučaju optimizacije globalnih problema sa ograničenjima FWA metaheuristika poseduje dovoljno jak mehanizam diversifikacije, ali to ostaje tema za naredna istraživanja.

5.4.3 Hibridizacija FWA sa metaheuristikom FA

Nedostatak metaheuristike FWA koji se manifestuje sporom konvergencijom i slabim procesom intenzifikacije može da se ispravi hibridizacijom. Za optimizaciju globalnih problema bez ograničenja, jaka diversifikacija pomoću Gausovih varnica nije potrebna i u slučaju ovih problema ovaj mehanizam samo bespotrebno troši računarske resurse.

S druge strane, ispitivanjem metaheuristike FA, pokazano je da ova metaheuristika koristi sofisticiranu jednačinu intenzifikacije koja se bazira na operatoru mutacije [144]. Zbog ovoga je predložena hibridizacija FWA i metaheuristika FA.

Predloženi hibrid FWA - pretraga svitaca (eng. fireworks algorithm - firefly search

- FWA-FS) je hibrid niskog nivoa, kod koga je komponenta pretrage metaheuristike FA integrisana u FWA [164]. Obzirom da je mehanizam Gausove eksplozije u slučaju problema globalne optimizacije bez ograničenja suvišan, on se jednostavno zamenjuje FA jednačinom pretrage (jednačina 5.29).

Integrisanjem FA jednačine pretrage u metaheuristiku FWA, pojačan je proces intenzifikacije tokom celog izvršavanja algoritma. Na taj način je i balans između intenzifikacije i diversifikacije pomeren u korist intenzifikacije.

I osnovni i hibridni FWA pristup već u početnim ciklusima lako pronalaze odgovarajući region prostora pretrage. Međutim, hibrid FWA-FS pojačanom intenzifikacijom vrši finiju pretragu u optimalnom domenu i na taj način se brzina konvergencije poboljšava u odnosu na originalnu verziju.

Novi mehanizam pretrage prikazan je pseudo-kodom u Algoritmu 20 [164]. U pseudo-kodu se predpostavlja da je $\hat{x}_s$ svetlija od varnice $\hat{x}_j$, a ϵ_j je psudo-slučajan broj iz intervala $[0, 1]$. Formulacija atraktivnosti ista je kao i u osnovom FA.

Algoritam 20 Pseudo-kod FWA-FS algoritma

odredi početnu lokaciju varnica $\hat{x}_j = x_i$
izaberi slučajnih z dimenzija iz $\hat{x}_j$ pomoću jednačine (5.74)
izračunaj osvetljenost svih varnica i pronađi varnicu koja je svetlija od $\hat{x}_j$
for svaku izabranu dimenziju $\hat{x}_k^j$ od $\hat{x}_j$ **do**
  $\hat{x}_k^j = \hat{x}_k^j + \beta_0 r^{-\gamma r_{k,s}^2}(\hat{x}_s^j - \hat{x}_k^j) + \alpha\epsilon_j$
  if $\hat{x}_k^j < x_k^{min}$ ili $\hat{x}_k^j > x_k^{max}$ **then**
    mapiraj $\hat{x}_k^j$ na prostor potencijalnih rešenja
  end if
end for

Početna vrednost parametra slučajnosti α FA jednačine pretrage se postepeno smanjuje sa svoje početne vrednosti tokom izvršavanja algoritma na osnovu sledećeg izraza:

$$\alpha(t) = (1 - (1 - ((10^{-4}/9)^{1/FE}))) \cdot \alpha(t-1), \tag{5.77}$$

gde je t brojač tekuće iteracije, a FE maksimalni broj evaluacija funkcija.

5.4.3.1 Eksperimentalni rezultati

Kako bi se izmerile performanse hibrida FWA-FS, izvršena su dva testa sa šest standardnih benčmark funkcija globalne optimizacije bez ograničenja: *Sphere, Rosenbrock, Rastrigin, Griewank, Schwefel* i *Ackley* [164]. Broj varijabli svih test funkcija je postavljen na 30 ($D = 30$).

Za merenje performansi algoritma, korišćena su dva indikatora - statistički prosek i standardna devijacija - rešenja problema koja su generisana. Za potrebe realnije komparativne analize, prosečne vrednosti su dobijene izvršavanem algoritma dvadeset puta, kao i u radu u kome su prikazani rezultati koji su korišćeni za komparativnu analizu [59].

U testovima su korišćene iste vrednosti osnovnih FWA parametara, kao i u [59]: $n = 5$, $m = 50$, $\alpha = 0.04$, $\beta = 0.8$, $\hat{A} = 40$ i $\hat{m} = 5$. Parametri koji su specifični za metaheuristiku FA postavljeni su na sledeće vrednosti: atraktivnost na udaljenosti $r = 0$ β_0 na 0.2, početna vrednost parametra slučajnosti α na 0.5 i koeficijent apsorpcije svetlosti γ na 1.0.

U tabelama sa rezultatima (Tabela 5.18 i Tabela 5.19), najbolji rezultati iz svih kategorija testova označeni su masnim slovima. Svi rezultati su zaokruženi na sedam decimala.

FWA-FS [164] je upoređivan sa originalnim FWA [59], klonirajućim PSO (eng. clonal PSO - CPSO) i standardnim PSO (eng. standard PSO - SPSO) [165]. U prvom testu je za svaki benčmark problem korišćen drugačiji broj evaluacija funkcija, i to: *Sphere* sa 500,000 evaluacija, *Rosenbrock* sa 600,000 evaluacija, *Rastrigin* sa 500,000 evaluacija, *Griewank* sa 200,000 evaluacija, *Schwefel* sa 600,000 evaluacija i *Ackley* sa 200,000 evaluacija funkcija. Rezultati prvog testa su prikazani u Tabeli 5.18.

Kao što se vidi iz rezultata prikazanih u Tabeli 5.18, i originalni i hibridni FWA postižu bolje rezultate od CPSO i SPSO u svim izvedenim testovima. Za sve test probleme, osim za *Rosenbrock*, i osnovna i hibridna metaheuristika uspele su da postignu optimume, i to bez diversifikacije rešenja u populaciji (standardna devijacija je 0).

Tabela 5.18: Rezultati testiranja sa različitim broje evaluacija funkcija

Problem		**CPSO**	**SPSO**	**FWA**	**FWA-FS**
Sphere	Najbolji	0.000000	1.909960	0.000000	0.000000
	StDev	0.000000	2.594634	0.000000	0.000000
Rosenbrock	Najbolji	33.40319	410.5225	9.569493	**6.002849**
	StDev	42.51345	529.3891	12.12829	**11.58233**
Rastrigin	Najbolji	0.053042	167.2561	0.000000	0.000000
	StDev	0.370687	42.91287	0.000000	0.000000
Griewank	Najbolji	0.632403	2.177754	0.000000	0.000000
	StDev	0.327648	42.91287	0.000000	0.000000
Schwefel	Najbolji	0.095099	0.335996	0.000000	0.000000
	StDev	0.376619	0.775270	0.000000	0.000000
Ackley	Najbolji	1.683649	12.36541	0.000000	0.000000
	StDev	1.317866	1.265322	0.000000	0.000000

Za problem *Rosenbrock*, hibrid FWA-FS postiže bolje rezultate od osnovnog FWA. U ovom slučaju jasno se vidi da je primena jednačine pretrage metaheuristike FA poboljšala brzinu konvergencije algoritma, a time i najbolje i prosečne rezultate. Najbolji rezultati koje generiše FWA-FS bolji su čak za 41 % u odnosu na osnovni FWA.

U drugom eksperimentu, vršena su testiranja svih funkcija sa 10,000 evaluacija funkcija. Rezultati drugog testa prikazani su u Tabeli 5.19.

Kao i u prvom testu, i hibridna i osnovna metaheuristika FWA pokazale su mnogo bolje performanse nego CPSO i SPSO i pronašle su optimalna rešenja za većinu benčmark problema za manje od 10,000 evaluacija funkcija. Međutim, u testovima sa 10,000 evaluacija, preciznost CPSO i SPSO nije zadovoljavajuća i algoritmi postižu rezultate koji su daleko od optimuma.

Hibrid FWA-FS postigao je bolje rezultate od osnovnog FWA za vrednost indikatora najboljeg rezultata u *Rosenbrock* i $Schwefel$ testovima. Međutim, originalni FWA postigao je bolje rezultate od FWA-FS kada se posmatra indikator standardna devijacija za *Rosenbrock* test.

Tabela 5.19: Rezultati testiranja sa 10,000 evaluacija funkcija

Problem		**CPSO**	**SPSO**	**FWA**	**FWA-FS**
Sphere	Najbolji	11857.42	24919.09	0.000000	0.000000
	StDev	3305.973	3383.241	0.000000	0.000000
Rosenbrock	Najbolji	2750997	5571942	19.38330	**16.90157**
	StDev	1741747	9604216	**11.94373**	12.78335
Rastrigin	Najbolji	10940.14	24013.00	0.000000	0.000000
	StDev	3663.484	4246.961	0.000000	0.000000
Griewank	Najbolji	3.457273	7.125976	0.000000	0.000000
	StDev	0.011027	0.965788	0.000000	0.000000
Schwefel	Najbolji	8775860	6743699	4.353733	**4.117829**
	StDev	1217609	597770.0	0.000000	0.000000
Ackley	Najbolji	15.90766	18.42334	0.000000	0.000000
	StDev	1.196082	0.503372	0.000000	0.000000

Ispitivanjem ponašanja algoritma u *Rosenbrock* testu, zaključeno je da je za njegovo rešavanje ipak potrebna snažnija diversifikacija, koja nedostaje u FWA-FS zbog eliminacije Gausovim varnica. Međutim, čak i u ovom slučaju unapređenja u odnosu na osnovni FWA su značajna, pošto se uz malo žrtvovanje standardne devijacije (pogoršanje od 6.6 %) generišu mnogo bolji najbolji rezultati (poboljšanje od 14.6 %).

5.4.4 Zaključak

FWA je metaheuristika koju su kreirali Tan i Zhu 2010. godine [59]. Ovaj algoritam modelira proces eksplozije vatrometne rakete.

FWA je ispitivana na problemima globalne optimizacije bez ograničenja. Ispitivanjem je utvrđeno da algoritam konvergira sporo ka optimumu zbog nedovoljno jakog procesa intenzifikacije. S druge strane, utvrđeno je da se za klasu ovakvih problema algoritam retko zaglavljuje u suboptimalnim domenima, i zbog toga, jaka diversifikacija koja se izvodi mehanizmom Gausovih varnica nije potrebna.

U osnovnom FWA pristupu ne postoji eksplicitna kontrola nad balansom između

intenzifikacije i diversifikacije, niti je jasno kada i u kojoj meri metaheuristika FWA izvodi intenzifikaciju, a kada diversifikaciju. U početnim iteracijama, diversifikacija bi trebala da bude jača, a kasnije bi trebala da se smanjuje u korist intenzifikacije. Međutim, u FWA pristupu se tokom celog izvršavanja algoritma procesi intenzifikacije i diversifikacije izvršavaju ravnomerno na osnovu vrednosti parametra χ.

U predloženom hibridnom pristupu, FWA-FS, umesto mehanizma Gausove varnice, koristi se FA jednačina pretrage [164]. Na taj način je diversifikacija smanjena, a intenzifikacija je značajno pojačana.

Algoritam je testiran na 6 standardnih benčmark funkcija za globalnu optimizaciju bez ograničenja. Izvršena je komparativna analiza sa osnovnim FWA i druge dve metaheuristike. Rezultati su potvrdili da je FWA-FS bolja metaheuristika od FWA algoritma. Takođe, u poređenju sa drugim algoritmima, FWA-FS postiže bolje performanse.

5.5 Hibridizacija metaheuristike BA

5.5.1 Osnovna metaheuristika BA

Algoritam slepog miša (eng. bat algorithm - BA) razvio je Yang 2010. godine i predložio ga za probleme globalne optimizacije sa ograničenjima varijabli [58]. Kasnije su predložene varijacije BA i za druge vrste problema, kao što su inženjerska optimizacija [74] i problemi sa više funkcija cilja [75].

BA modelira mogućnost eholokacije slepih miševa koji koriste zvučne odjeke za detektovanje i izbegavanje prepreka. Slepi miševi za navigaciju u prostoru koriste vreme koje protekne od emitovanja do reflektovanja zvuka o prepreke i druge objekte iz okruženja.

Inspirisan karakteristikama slepih miševa, BA se temelji na sledećim pojednostavljenim pravilima, koja olakšavaju modeliranje realnog sistema[58]:

- slepi miševi imaju osećaj za razdaljinu pomoću eholokacije i mogu da odrede koliko su udaljeni od hrane, tj. plena i prepreka u okruženju;

- u potrazi za potencijalnim plenom slepi miševi lete slučajnim letom brzine v_i, a pritom imaju sledeće karakteristike: poziciju x_i, frekvenciju f_i sa varirajućom talasnom dužinom λ i jačinu zvuka A_0. Slepi miševi mogu samostalno da podešavaju frekvenciju stope pulsiranja zvuka $r \in [0, 1]$, koja zavisi od toga koliko su slepi miševi udaljeni od potencijalnog plena i
- pretpostavka je da jačina zvuka varira sa najveće početne vrednosti A_0 na minimalnu konstantnu vrednost, koja se označava sa A_{min}.

Osim svih navedenih pravila, originalni BA koristi i pojednostavljenje da se frekvencija f nalazi u opsegu $[f_{min}, f_{max}]$ i da odgovara opsegu talasnih dužina $[\lambda_{min}, \lambda_{max}]$. Na primer, opseg frekvencija $[20, 500]$ kHz odgovara opsegu talasnih dužina od $[0.7, 17]$ mm.

Radi daljeg uprošćavanja realnog sistema, predpostavlja se da $f \in [0, f_{max}]$. Poznato je da veće frekvencije imaju kraće talasne dužine i da prelaze kraće razdaljine. Tipične talasne dužine za slepe miševe su do nekoliko metara. Stopa pulsiranja jednostavno može da se definiše u opsegu $[0, 1]$, gde 0 označava da nema pulsiranja, a 1 označava najveću stopu pulsiranja.

Svaki slepi miš u populaciji je određen svojom brzinom v_i^t i pozicijom x_i^t u iteraciji t u d-dimenzionom prostoru pretrage. Kretanje slepog miša modelira se ažuriranjem njegove brzine i pozicije u svakoj iteraciji pomoću sledećeg izraza [58]:

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{5.78}$$

$$v_i^t = v_i^{t-1} + (x_* - x_i^{t-1})f_i, \tag{5.79}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{5.80}$$

gde je f_i vrednost frekvencije slepog miša i, a $\beta \in [0, 1]$ je pseudo-slučajan broj generisan uniformnom distribucijom. Parametar x_* je trenutno najbolje rešenje (slepi miš) koje se u svakoj iteraciji ponovo određuje.

U originalnom BA pristupu [58], vrednosti za f_{min} i f_{max} su postavljene na 0 i 100, respektivno. U fazi inicijalizacije, za svakog slepog miša u populaciji generiše se slučajna frekvencija u intervalu [f_{min},f_{max}].

Primenom procedure lokalne pretrage za svakog slepog miša se generiše novo rešenje, koje se nalazi u okolini trenutne pozicije, pomoću slučajnog koraka [58]:

$$x_{new} = x_{old} + \epsilon A^t, \tag{5.81}$$

gde ϵ označava slučajan vektor iz intervala $[-1, 1]$, a $A^t =< A_t^i >$ označava prosečnu jačinu zvuka svih slepih miševa u populaciji u određenom vremenskom trenutku.

Jačina zvuka A_i i stopa emitovanja impulsa r_i takođe se ažuriraju tokom svih ciklusa izvršavanja algoritma. Kada slepi miš pronađe svoj plen, njegova jačina zvuka se obično smanjuje, dok se stopa emitovanja pulsa povećava. Ovo se modelira sledećim izrazom:

$$A_i^t = \alpha A_i^{t-1},\ r_i^t = r_i^0\ [1 - \exp(-\gamma t)], \tag{5.82}$$

gde su α i γ konstante. Za svako α u intervalu $(0, 1)$, i $\gamma > 0$, važi da [58]:

$$A_i^t \to 0,\ r_i^t \to r_i^0,\ \text{dok } t \to \infty \tag{5.83}$$

U najjednostavnijem slučaju takođe može da se koristi jednakost $\alpha = \gamma$, a u originalnom radu korišćene su vrednosti $\alpha = \gamma = 0.9$ [58].

Pseudo-kod originalne metaheuristike BA prikazan je u Algoritmu 21. U navedenom pseudo-kodu IN označava broj ciklusa u toku jednog izvršavanja algoritma.

Algoritam 21 Pseudo-kod metaheuristike BA

definiši funkciju cilja $f(x),\ x = (x_1, x_2, ..., x_d)^T$
inicijalizuj popuilaciju slepih miševa $x_i,\ (i = 1, 2, ...n)$ i v_i
definiši frekvenciju impulsa f_i na poziciji x_i
inicijalizuj stope emitovanja impulsa r_i i jačinu zvuka A_i
while $t < IN$ **do**
  generiši nova rešenja podešavanjem frekvencije i ažuriranjem brzina i pozicija
  rešenja primenom jednačina (5.78) - (5.80)
  if $rand > r_i$ **then**
    izaberi najbolje rešenje iz populacije
    generiši novo rešenje u okolini izabranog rešenja
  end if
  generiši novo rešenje slučajnim letom
  if $rand < A_i$ i $f(x_i) < f(x_*)$ **then**
    prihvati nova rešenja
    povećaj r_i i smanji A_i
  end if
  rangiraj sve slepe miševe u populaciji i pronađi trenutno najbolje rešenje x_*
end while
obradi sve rezultate i vizualizuj najbolja rešenja

5.5.2 Nedostaci metaheuristike BA

Metaheuristika BA je ispitivana za probleme globalne optimizacije bez ograničenja. Ovaj algoritam je jedini pristup iz familije metaheuristika rojeva koji koristi mehanizam podešavanja frekvencije. Osnovna uloga ovog mehanizma je izvođenje procesa intenzifikacije pomoću operatora mutacije.

Jednačine koje modeliraju kretanje individua u populaciji (5.78) - (5.80) ne primenjuju eksplicitno operator ukrštanja. Postoje i primeri drugih metaheuristika rojeva koje eksplicitno ne koriste ovaj operator, kao što su ABC i PSO.

Međutim, s obzirom da proces mutacije menja intezitet tokom izvršavanja algoritma, i pored nepostojanja operatora ukrštanja, proces intenzifikacije BA je relativno dobar. Intenzitet mutacije menja se promenama jačine zvuka (A_i) i stope emitovanja pulsa (r_i) i na taj način se, tokom izvršavanja algoritma, pretraga sve više usmerava

ka najboljim rešenjima u populaciji. Pomoću ovih parametara uspostavlja se balans između eksploatacije i eksploracije.

Selekcioni pritisak je relativno nepromenljiv tokom izvršavanja algoritma, zato što se individue usmeravaju ka trenutno najboljem rešenju x_* pomoću jednačine (5.79). Međutim, ovo usmeravanje može da ima dvostruki efekat. S jedne strane, ovakav mehanizam je dobar u kasnijim iteracijama, kada je algoritam konvergirao ka optimalnom regionu. U tom slučaju se usmeravanjem ka najboljim rešenjima izvodi fina pretraga optimalnog regiona i proces pretrage brzo konvergira ka optimumu. S druge strane, u ranijim iteracijama, pretraga orijentisana ka najboljim rešenjima može da dovede do preuranjene konvergencije, i algoritam pokazuje lošije prosečne rezultate.

Ovaj problem može da se posmatra i s aspekta slabije diversifikacije. Ispitivanjem je utvrđeno da BA ponekada u ranijim iteracijama, zbog usmeravanja ka trenutno najboljem rešenju, ne uspeva da pogodi pravi deo prostora pretrage. Ovaj problem može da se ukloni pojačanjem diversifikacije, ili balansiranjem sa nekom drugom metodom koja se ne orijentiše ka trenutno najboljem rešenju.

Generalno, BA pokazuje dobre rezultate za probleme globalne optimzacije bez ograničenja, ali postoji prostor za poboljšanja. Kao dva nedostatka (uz napomenu da su nedostaci slabo izraženi) metaheuristike BA navodi se problem preuranjene konvergencije zbog usmerene pretrage ka trenutno najboljem rešenju i problem slabije divesifikacije u ranijim ciklusima izvršavanja algoritma.

5.5.3 Hibridizacija BA sa metaheuristikom ABC

Jedan od načina da se ispravi nedostatak preuranjene konverencije metaheuristike BA jeste balansiranje sa mehanizmom koji ne koristi pretragu koja se usmerava ka trenutno najboljem rešenju u populaciji.

Predložena hibridizacija sa ABC algoritmom, BA sa mehanizmom posmatrača (eng. BA with onlooker mechanism - BA-OM), adaptira proceduru pčela posmatrača iz ABC algoritma. Pčele posmatrači koriste probabilističku selekciju na osnovu po-

dobnosti individau u populaciji [166]. BA-OM je adaptiran za benčmark probleme globalne optimizacije bez ograničenja i za praktičan problem planiranja RFID mreže.

U prvom koraku BA-OM generiše početnu populaciju od N slučajno distribuiranih rešenja, gde je svako rešenje vektor od k dimenzija (varijabli). Cela populacija se prikazuje kao matrica X:

$$X_{1,1}, X_{1,2}, ..., X_{2,1}, X_{2,2}, ..., X_{N-1,k-1} \tag{5.84}$$

Svaka individua (slepi miš) u poplaciji generiše se pomoću jednačine:

$$x_{i,j} = lb_j + \phi * (ub_j - lb_j), \tag{5.85}$$

gde je $x_{i,j}$ j-ta varijabla i-tog rešenja, ϕ je pseudo-slučajan realan broj između 0 i 1, a ub_j i lb_j su gornja i donja granica j-te varijable, respektivno.

Jednačina (5.85) takođe se koristi za izračunavanje brzine $v_{i,j}$. U ovom koraku se takođe brojač trenutnih ciklusa postavlja na 0, računa se vrednost funkcije cilja za sve individue u populaciji i određuje se najbolje rešnje x_*.

U drugom koraku BA-OM izvodi proces intenzifikacije. Ovde se osim osnovnih jednačina pretrage originalnog BA (jednačine (5.78) - (5.80)) koristi i procedura pretrage pčela posmatrača iz metaheuristike ABC. Vrednosti najmanje i najveće frekvencije su postavljene redom na $f_{min} = 0$ i $f_{max} = 2$.

Pčela posmatrač ABC algoritma bira rešenje za eksploataciju u sledećoj iteraciji sa verovatnoćom koja je proporcionalna podobnosti rešenja [66]. BA-OM koristi sličan princip. Prvo se za sve individue u populaciji računa vrednost funkcije podobnosti fit_i jednačinom:

$$fit_i = \begin{cases} \frac{1}{1+f_i}, & \text{ako je } f_i \geq 0 \\ 1 + |f_i|, & \text{u suprotnom,} \end{cases} \tag{5.86}$$

gde je f_i funkcija cilja i-tog slepog miša u populaciji.

Nakon toga se računa verovatnoća izbora rešenja primenom jednačine:

$$p_i = \frac{fit_{avg}}{\sum_{i=1}^{m} fit_i}, \tag{5.87}$$

gde je p_i verovatnoća da će i-to rešenje biti izabrano, dok je fit_{avg} prosečna podobnost individua u populaciji. Dakle, koristi se jednostavna rulet selekcija.

Izabrano rešenje i se eksploatiše prema osnovnoj jednačini pčele posmatrača [66]:

$$x_{i,j} = x_{i,j} + \phi * (x_{i,j} - x_{k,j}), \tag{5.88}$$

gde je $x_{i,j}$ j-ti parametar starog rešenja i, $x_{k,j}$ je j-ti parametar susednog rešenja k, a ϕ je pseudo slučajan broj iz intervala $[0, 1]$. BA-OM ne koristi MR parametar kao ABC algoritam za optimizaciju sa ograničenjima [66].

Ako je kvalitet novog rešenja veći od kvaliteta starog, novo rešenje se zadržava, i obrnuto.

Da bi se kontrolisalo da li će mehanizam pčele posmatrača da se koristi u tekućoj iteraciji, koristi se brojač trenutne iteracije. Ispitivanjem je utvrđeno da se najbolji balans postiže ako se procesi pretrage BA i pčela posmatrača smenjuju, i zato se u nekim iteracijama primenjuje mehanizam pretrage metaheuristike BA, a u nekim mehanizam pčele posmatača. Na ovaj način, vrši se dinamičko prebacivanje sa jednog na drugi mehanizam pretrage u zavisnosti od balansa između intenzifikacije i diversfikacije u tekućoj iteraciji.

Konačno, u trećem koraku koristi se procedura lokalne pretrage. Novo rešenje za svakog slepog miša generiše se u okolini trenutno najboljeg rešenja pomoću slučajnog koraka (jednačina 5.81).

Pseudo-kod metaheuristike BA-OM prikazan je u Algoritmu 22 [166]. Parametar IN označava maksimalan broj iteracija i predstavlja kriterijum za završetak jednog izvršavanja algoritma.

Algoritam 22 Pseudo-kod metaheuristike BA-OM

definiši funkciju cilja $f(x), x = (x_1, x_2, ..., x_d)^T$
inicijalizuj populaciju slepih miševa
$X_{i,j}(i = 1, 2, 3, ...N)(j = 1, 2, 3, ...D)$
definiši frekvenciju impulsa f_i na poziciji x_i
inicijalizuj stope emitovanja impulsa r_i i jačinu zvuka A_i
brojač trenutne iteracije t postavi na 0
while $t < IN$ **do**
  if $t\%2 == 0$ **then**
    generiši nova rešenja primenom BA pretrage pomoću (5.78) - (5.80)
  else
    primeni proceduru pretrage pčele posmatrača, jednačina (5.88)
  end if
  if $rand > r_i$ **then**
    izaberi slučajna rešenja između najboljih rešenja
    generiši lokalno rešenje slučajnim letom pomoću (5.81)
  end if
  if $rand < A_i$ i $f(x_i) < f(x_*)$ **then**
    prihvati nova rešenja
    povećaj r_i i smanji A_i
  end if
  rangiraj sve slepe miševe u populaciji i pronađi trenutno najbolje rešenje x_*
end while
obradi sve rezultate i vizualizuj najbolja rešenja

5.5.3.1 Eksperimentalni rezultati

Hibridna metaheuristika BA-OM prvo je testirana na pet standardnih funkcija globalne optimizacije bez ograničenja [166]. U drugom setu eksperimenata vršena su testiranja na praktičnom problemu planiranja RFID mreže. Da bi se izmerilo unapređenje performansi hibridnog algoritma, implementiran je i osnovni BA za iste probleme.

Na problemima globalne optimizacije bez ograničenja, BA-OM je testiran u 50 nezavisnih izvršavanja sa 100,000 ciklusa u jednom izvršavanju i veličinom populacije od 100 individua [166]. Isti parametri korišćeni su i za testiranje drugih metaheuristika,

čiji su rezultati preuzeti za potrebe komparativne analize [141].

Podešavanja parametara metaheuristike BA-OM sumirana su u Tabeli 5.20.

Tabela 5.20: Podešavanja BA-OM parametara

Parametar	**Vrednost**
Broj individua (n)	100
Broj iteracija	10.000
Jačina zvuka A_0	100
Jačina zvuka A_{min}	1
f_{min}	0
f_{max}	1
Konstanta α	0.9
Konstanta γ	0.9

Funkcije globalne optimizacije koje su korišćene za potrebe testiranja dobro su poznate u literaturi koja se bavi metaheuristikama rojeva [167]. Prvi problem je dobro poznat *Sphere* benčmark koji ima jedan optimum, *Rosenbrok*, *Grienwank* i *Weierstrass* problemi imaju više optimuma, dok poslednji benčmark ima naziv *Composition Function 1.* U testiranjima su korišćene funkcije sa 30 varijabli.

BA-OM je upoređivan sa sešt drugih metaheuristika koje su sve testirane na istim problemima i pod istim uslovima. U komparativnu analizu uvršćeni su: PS^2O (eng. multi-swarm particle swarm optimizer) [141], CPSO (eng. PSO with constriction factor) [168], FIPS (eng. fully informed particle swarm) [169], UPSO (eng. unified particle swarm) [170], FDR-PSO (eng. fitness-distance-ratio based PSO) [171] i MCPSO (eng. multi-swarm cooperative PSO) [172]. Da bi se izmerila prava unapređenja koja generiše BA-OM, implementiran je i osnovni BA za isti skup problema. U analizu je uključen i ABC [173], pošto se on koristio za hibridizaciju.

Rezultati komparativne analize prikazani su u Tabeli 5.21. Najbolji rezultati označeni su masnim slovima radi bolje preglednosti.

Tabela 5.21: Poređenje BA-OM za benčmark probleme

Prob.	PS^2**0**	**CPSO**	**FIPS**	**UPSO**	**FDR-PSO**	**MCPSO**	**ABC**	**BA**	**BA-OM**
Sphere									
Najbolji	0	2.4787e-116	1.1874e-30	3.4459e-185	6.9665e-190	7.5976e-39	-	0	0
Najgori	0	1.3486e-113	9.7762e-029	1.9929e-182	7.4365e-168	3.8011e-32	-	0	0
Prosečni	0	2.4205e-114	1.7391e-29	3.7072e-183	2.4789e-169	4.5248e-33	7.57e-04	0	0
Std.	0	3.3966e-114	2.2995e-029	0	0	1.1842e-32	2.48e-04	0	0
Rosenbrok									
Najbolji	1.5203e-15	5.8889	17.4217	0.7070	0.0012	0.1394	-	8.75e-11	**3.29e-16**
Najgori	**5.1336e-14**	7.4375	23.4450	4.0368	4.0879	13.7541	-	3.03e-9	1.05e-13
Prosečni	1.0412e-14	6.6172	22.5407	2.0983	0.2797	4.7876	9.36e-01	7.31e-10	**3.55e-15**
Std.	9.7087e-15	0.4028	1.2748	0.7696	1.0224	4.6528	1.76e+00	5.18e-11	**4.29e-16**
Grienwank									
Najboljit	0	0	0	0	0	0	-	0	0
Najgori	0	0.1152	0.0123	0.0388	0.0737	0.0346	-	9.65e-72	0
Prosečni	0	0.0183	0.0016	0.0347	0.0179	0.0059	8.37e-04	4.54e-102	0
Std.	0	0.0266	0.0038	0.0478	0.0182	0.0111	1.38e-03	0.32e-110	0
Weierstrass									
Najbolji	0	2.8242e-5	0	0	0	4.2985	-	0	0
Najgori	0	3.7591	0.2856	8.1054	1.5086	13.2105	-	0	0
Prosečni	0	1.3510	0.0201	4.4244	0	0	-	0	0
Std.	0	70.7121	25.8645	0.0143	0.1581	7.6180	-	0	0
Comp. Fun. 1									
Najbolji	0	0.0051	0	0	0	1.4462e-30	-	0	0
Najgori	0	100.0071	45.5672	0.0467	300.00	3.7882e-28	-	5.98e-60	0
Prosečni	0	50.0061	33.7051	0.0136	100.00	2.3917e-28	-	7.02e-85	0
Std.	0	70.7121	25.8645	0.0143	141.42	2.6684e-28	-	6.25e-93	0

Tabela 5.21, pokazuje da PS^2O, BA i BA-OM ispoljavaju odlične performanse za četiri od pet benčmark problema. Najznačajnija razlika vidi se u *Rosenbrock* testu. U ovom slučaju BA-OM postiže bolje rezultate od PS^2O i BA. Samo indikator najgoreg rezultata PS^2O ima bolju vrednost nego BA-OM.

Razlika u performansama takođe se vidi i u *Composite function 1* testu, gde BA-OM postiže bolje rezultate za sve indikatore performansi. Ostali algoritmi, CPSO, FIPS i UPSO za sve test probleme postižu značajno lošije rezultate.

U drugom setu eksperimenata, BA-OM je testiran na praktičnom problemu planiranja RFID mreže. Korišćena je slična formulacija ovog problema kao i u testovima hibridne metaheuristike ABC [136], [137].

Predložen problem RFID mrežnog planiranja pripada grupi teške višekriterijumske optimizacije, gde je potrebno simultano optimizovati sledeće ciljeve: optimalna pokrivenost tagova, izbegavanje kolizije čitača, ekonomska efikasnost i balansiranje opterećenja.

Prvi i najvažniji cilj je *optimalna pokrivenost tagova.* Komunikacija između čitača i taga može da se uspostavi samo ako je snaga radio signala koju prima tag veća od vrednosti praga osetljivosti $P_d = -10dBm$. Optimalna pokrivenost tagova formuliše se kao suma razlike između željenog nivoa snage P_d i snage P_i^r koju prima svaki tag i [141]:

$$\min\ C = \sum_{i=1}^{N_T} (P_i^r - P_d), \tag{5.89}$$

gde je N_T broj tagova u domenu mreže.

Drugi cilj, *izbegavanje kolizije čitača*, definiše se na sledeći način:

$$\min\ I = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} (r_i + r_j) - (\text{dist}(R_i, R_j), \tag{5.90}$$

gde je M broj čitača, dist() je funkcija koja računa udaljenost između čitača, R_i i R_j su pozicije čitača, a r_i i r_j je domet interferencije između čitača i i j, respektivno.

Cilj *ekonomska efikasnost* osigurava da su čitači blizu centara klastera tagova:

$$\min\ E = \sum_{k=1}^{M}(dist(R_k - Center_k)), \tag{5.91}$$

gde je M broj čitača, $dist()$ je udaljenost između k-tog čitača i centra k-tog klastera, dok su $Center_k$ i R_k pozicije centra k-tog klastera i njemu najbližeg čitača, respektivno.

Četvrti cilj, *balansiranje opterećenja* modelira zahtev da je potrebno omogućiti što veću pouzdanost RFID sistema:

$$\min\ L = \prod_{k=1}^{M}(\frac{1}{C_k}), \tag{5.92}$$

gde je C_k broj tagova koje opslužuje čitač k u okviru domena mreže.

Da bi se optimizovao problem planiranja RFID mreže, algoritam mora da se adaptira, tako što se svaka individua predstavlja pomoću $3M$ dimenzija, gde je M broj korišćenih RFID čitača. Prve dve dimenzije kodiraju koordinate čitača, a poslednja kodira emitovanu snagu.

Algoritam je testiran u mrežnom domenu veličine $30 * 30m$ sa 100 tagova koji su uniformno distribuirani. Za opsluživanje mreže koristi se 10 čitača. Ovako postavljen problem pripada grupi problema globalne neprekidne optimizacije sa 30 varijabli. Tehničke specifikacije problema prikazane su u Tabeli 5.22.

Tabela 5.22: Tehničke specifikacije RFID problema

Spec. čitača i topologije	**Vrednost**
Broj čitača	10
Opseg ispitivanja	3-4m
Opseg interferencije	3.5-4.5m
Broj tagova	100
Distribucija tagova	uniformna
Prag snage taga	-10dBm

Svi eksperimenti su vršeni sa 2,000 iteracija i veličinom populacije od 50 jedinki. Algoritmi koji su uključeni u komparativnu analizu prvo su testirani posebno za

svaki cilj RFID problema: optimalna pokrivenost tagova (C), izbegavanje kolizije čitača (I), ekonomska efikasnost (E) i balansiranje opterećenja (L). Nakon toga je optimizovana kombinovana funkcija cilja (A), koja se izvodi sledećom jednačinom [141]:

$$\min A = \sum_{i=1}^{4} \omega_i f_i / f_{i,max}, \quad \omega_1 + \omega_2 + \omega_3 + \omega_4 = 1, \ \omega_i > 0, \tag{5.93}$$

gde je f_i funkcija cilja i-tog zahteva normalizovana na svoju najveću vrednost $f_{i,max}$.

Težinski koeficijenti ω su postavljeni, redom za svaki cilj, na sledeće vrednosti: $\omega_1 = 0.4$, $\omega_2 = 0.2$, $\omega_3 = 0.2$ i $\omega_4 = 0.2$ [141]. Algoritam je za svaki test izvršavan 50 puta, kako bi se utvrdile prosečne vrednosti.

Komparativnom analizom obuhvaćeni su sledeći algoritmi: PS^2O [141], CPSO [168], MCPSO [141], SA-ES [141] i EGA [141]. Analiza je prikazana u Tabeli 5.23.

Na osnovu rezultata iz prikazane tabele, zaključuje se da BA-OM postiže zadovoljavajuće performanse u rešavanju problema planiranja RFID mreže. U optimizaciji cilja pokrivenost tagova (C), BA-OM postiže slične rezultate kao PS^2O, koji su bolji od rezultata ostalih algoritama. PS^2O postiže bolje performanse za indikator standardne devijacije, dok je BA-OM bolji u svim drugim slučajevima.

Takođe, i u optimizaciji cilja izbegavanje kolizije čitača (I), BA-OM postiže odlične rezultate, gde za indikatore najboljeg rezultata i standardnu devijaciju nadmašuje PS^2O.

U optimizaciji kriterijuma ekonomske efikasnosti (E), BA-OM postiže bolje performanse od svih drugih algoritama. Optimizaciju cilja balansiranje opterećenja (L), BA-OM postiže slične rezultate kao PS^2O.

Konačno, u kombinovanom testu (A), BA-OM je u proseku bolji od PS^2O zato što postiže mnogo bolje prosečne vrednosti i standardnu devijaciju. Za indikator najboljih rezultata, samo PS^2O i MCPSO postižu bolje vrednosti od predložene hibridizacije.

Tabela 5.23: Rezultati komparativne analize za RFID problem

Cilj	PS^2**0**	**CPSO**	**EGA**	**SA-ES**	**MCPSO**	**BA-OM**
C						
Najbolji	520.0115	538.9030	546.4006	618.1919	631.7577	**503.0408**
Najgori	733.5187	894.8067	940.1571	1.7747e+003	864.5434	**694.3859**
Prosečni	665.9523	756.7850	753.1622	1.1474e+003	723.2234	**612.5831**
StDev.	**63.0878**	114.7312	122.6468	331.7691	73.5192	69.0272
I						
Najbolji	324.4194	323.5060	374.0674	293.9330	327.1281	**291.3741**
Najgori	**333.8609**	369.0065	451.1235	356.6901	339.1420	338.9467
Prosečni	331.0160	335.2516	399.4469	332.0981	334.5781	**295.7842**
StDev	**2.9651**	13.1858	23.6057	15.6403	3.4705	15.1458
E						
Najbolji	1.1691e-10	1.1594e-10	1.5906e-10	1.4433e-10	1.5202e-10	**0**
Najgori	1.9290e-010	3.1124e-10	2.3239e-10	2.2596e-9	2.5476e-10	**0**
Prosečni	1.6131e-10	1.8593e-10	1.9425e-10	4.2330e-10	1.9679e-10	**0**
StDev	2.2053e-11	4.9145e-11	2.5148e-11	4.9474e-10	3.6410e-11	**0**
L						
Najbolji	0	5.2401e-9	2.2319	2.7861e-14	1.7135e-4	0
Najgori	0	3.2467e-8	16.2483	15.5788	0.0039	0
Prosečni	0	1.3935e-8	8.7723	2.9694	0.0011	0
StDev	0	6.6902e-009	3.8279	4.1988	0.0012	0
A						
Najbolji	**0.1798**	0.1994	0.2123	0.2351	0.1864	0.1853
Najgori	0.2254	0.2411	0.2758	0.3566	0.2429	**0.2239**
Prosečni	0.1980	0.2118	0.2394	0.2871	0.2105	**0.1967**
StDev	**0.1314**	0.1441	0.2173	0.3720	0.1556	0.1695

5.5.4 Zaključak

BA je relativno novija metaheuristika roja, koju je konstruisao Yang [58]. Jednačina pretraga BA ne koristi eksplicitno operator ukrštanja, ali se to nadomešćuje primenom mutacije, čiji se intenzitet menja tokom izvršavanja algoritma. U ovom radu je BA ispitivan za globalne probleme bez ograničenja.

Međutim, jednačina pretrage, koja je orijentisana ka najboljim rešenjima u popu-

laciji, ponekada izaziva probleme. U ranijim iteracijama, ovakva pretraga može da dovede do problema preuranjene konvergencije, i algoritam pokazuje lošije prosečne rezultate. Zbog ovoga se takođe dešava da algoritam u ranijim iteracijama ne uspeva da pronađe optimalni region.

Generalno, BA pokazuje dobre rezultate za probleme globalne optimzacije bez ograničenja, ali postoji prostor za poboljšanja. Kao dva nedostatka metaheuristike BA navodi se problem preuranjene konvergencije zbog usmerene pretrage ka trenutno najboljem rešenju i problem slabije divesifikacije u ranijim ciklusima izvršavanja algoritma.

Navedeni problemi mogu da se isprave pojačanjem diversifikacije, ili primenom metoda koji pretragu ne orijentiše ka trenutno najboljoj jedinki u populaciji. Sa ciljem poboljšanja BA, predložena je hibridizacija sa ABC algoritmom.

Predloženi hibrid, BA-OM, koristi mehanizam pčela posmatrača iz metaheuristike ABC, čime se orijentisana pretraga ka najboljem rešenju, balansira sa probabilističkim procesom selekcije na osnovu podobnosti individua [166]. Hibrid je testiran na benčmark problemima globalne optimizacije bez ograničenja i na praktičnoj primeni planiranja RFID mreže.

Utvrđeno je da BA-OM ispravlja nedostatke osnovnog BA i da postiže zadovoljavajuće performanse, kako na benčmark problemima, tako i na praktičnom problemu planiranja RFID mreže.

Zaključak

U ovom radu je predloženo unapređenje hibridizacijom nekih metaheuristika inteligencije rojeva za rešavanje problema globalne optimizacije. Oblast kojom se ovaj rad bavi predstavlja aktuelan problem iz oblasti rešavanja teških optimizacionih problema i rad se oslanja na najnovija istraživanja u svetu.

Detaljno je ispitivano pet algoritama inteligencije rojeva i predloženo je sedam hibirdizacija. Algoritmi su testirani na način koji je uobičajen u literaturi radi upoređivanja sa ostalim vodećim algoritmima: na standardnim skupovima brojnih i raznovrsnih test funkcija, sa istim broje evaluacija funkcije cilja, ali i primenama na pojedine važne i u literaturi zastupljene praktične probleme.

Ispitivanjem algoritama rojeva uočeni su u određenim slučajevima nedostaci i slabosti koji se pre svega odnose na proces pretrage i na neodgovarajući balans između intenzifikacije i diversifikacije. Zbog toga metaheuristike rojeva često generišu suboptimalna rešenja i zaglavljuju se u lokalnim optimumima.

Da bi se navedeni nedostaci korigovali, predložena je hibridizacija, kao ciljani i pažljivo vođen proces koji se temelji na sveobuhvatnom izučavanju načina na koji algoritmi koji se hibridizuju funkcionišu. Testiranjem na relevantnim instancama utvrđene su prednosti i nedostaci svakog algoritma koji je hibridizovan, kako bi se nedostaci jednog ispravili prednostima drugog pristupa.

U ovom radu su predložene hibridizacije sledećih metaheuristika rojeva: veštačke kolonije pčela (eng. artificial bee colony - ABC), optimizacije pretragom svitaca (eng. firefly algorithm - FA), optimizacije pretragom tragača (eng. seeker optimization algorithm - SOA), algoritma vatrometa (eng. fireworks algorithm - FWA) i algoritma slepog miša (bat algorithm - BA).

Uočeno je da kod metaheuristike ABC proces intenzifikacije nije dovoljno jak i da algoritam sporo konvergira ka optimalnom domenu prostora pretrage, posebno kod problema sa ograničenjima. Jedan od razloga je taj što osnovna ABC verzija koristi jednačinu pretrage koja se bazira na operatoru mutacije, bez eksplicitne primene

operatora ukrštanja. Takođe, uočeno je i da ABC ne uspostavlja odgovarajući balans između eksploatacije i eksploracije. Ovaj problem je izražen je i u ranijim ciklusima, ali je ozbiljniji u kasnijim fazama, kada je potrebno da se vrši fina pretraga optimalnog domena.

Radi prevazilaženja problema metaheuristike ABC, predložene su dve hibridizacije. Hibrid ABC i genetskih operatora (eng. genetically inspired artificial bee colony - GI-ABC) integriše genetske operatore uniformnog ukrštanja i mutacije [130]. Novi mehanizam, vođeni posmatrač, u kasnijim ciklusima zamenjuje mehanizam pčele izviđača. Na ovaj način GI-ABC rešava problem neusklađenog balansa između intenzifikacije i diversifikacije u kasnijim iteracijama. Algoritam je testiran na standardnom skupu benčmark problema sa ograničenjima [130] i na praktičnom problemu planiranja RFID mreže [136].

Druga predložena hibridizacija na instancama na kojima je testirana rešava problem slabije intenzifikacije i neodgovarajućeg balansa između eksploatacije i eksploracije tako što algoritam ABC kombinuje sa metaheuristikom FA (eng. ABC - firefly search - ABC-FS) [144]. Jednačina pretrage FA algoritma inkorporirana je u mehanizam pčele radilica metaheuristike ABC. ABC-FS je adaptiran za praktičan problem optimizacije portfolija sa ograničenjima kardinalnosti.

Suprotno od algoritma ABC, ispitivanjem je utvrđeno da metaheuristika FA za probleme sa ograničenjima ispoljava nedovoljnu diversifikaciju i da se algoritam često zaglavljuje u predelu lokalnih optimuma. Zbog toga je i balans između eksploatacije i eksploracije nedovoljno usklađen tokom celog izvršavanja algoritma.

Dve predložene hibridizacije metaheuristike FA potvrđuju da se pomenuti nedostaci mogu ispraviti hibridizacijom. Oba hibrida koriste mehanizam pčele izviđača metaheuristike ABC koji izvodi snažnu diversifikaciju. Prvi predloženi hibrid, unapređeni FA (eng. enhanced FA - eFA), prilagođen je za rešavanje problema sa ograničenjima, dok je drugi - modifikovani FA (eng. modified FA - mFA) - adaptiran za praktičan problem optimizacije portfolija sa ograničenjima kardinalnosti i entropije [147]. Iako elementi mehaheuristika FA i ABC pokazuju strukturnu kompatibilnost, bilo je potrebno da se pažljivim balansiranje vrednosti parametara, sinhronizuje jedan

proces pretrage sa drugim.

Za razliku od metaheuristika ABC i FA, SOA pokazuje slabosti i u procesu eksploatacije i u procesu diversifikacije. Prvo, ispitivanjem je utvrđeno da SOA nema naglašen mehanizam intenzifikacije, zato što se jednačina pretrage usmerava samo metodom fazi zaključivanja. Drugo, eksploracija koja se izvodi zamenom potpunih rešenja između podpopulacija daje rezultate samo u ranijim iteracijama, kada algoritam još uvek nije pronašao region optimalnog rešenja. Međutim, u kasnijim ciklusima, ovaj mehanizam donosi više štete nego koristi, zato što proces pretrage udaljava od optimalnog rešenja. Ponekada se čak dešava da zamena rešenja između podpopulacija dovede do preuranjene konvergencije.

Predloženi hibrid, SOA - pretraga svitaca (eng. SOA-firefly search - SOA-FS), prilagođen je za benčmark probleme sa ograničenjima [146]. Algoritam pomoću dodatnog kontrolnog parametra vrši usklađivanje jednačina pretraga metaheuristika SOA i FA, čime se proces intenzifikacije pojačava. Takođe, SOA-FS implementira dinamičku proceduru učenja između podpopulacija inkorporiranjem genetskog operatora ukrštanja. Novi, dinamički mehanizam predupređuje preuranjenu konvergenciju, i usmerava pretragu u kasnijim iteracijama na finu optimizaciju u okolini optimuma.

Slično kao i ABC algoritam, FWA ima jak mehanizam diversifikacije koji se implementira pomoću Gausovih varnica. Međutim, ispitivanjem metaheuristike FWA za optimizaciju globalnih problema sa ograničenjima vrednosti varijabli, utvrđeno je da ova procedura nije potrebna, pošto algoritam lako pronalazi optimalni region i retko se zaglavljuje u predelu suboptimalnih rešenja. Takođe, ustanovljeno je da u osnovnom FWA pristupu ne postoji eksplicitna kontrola nad balansom između intenzifikacije i diversifikacije, niti je jasno kada i u kojoj meri FWA jednačina pretrage izvodi eksploataciju, a kada eksploraciju.

U predloženom hibridnom pristupu, FWA-pretraga svitaca (eng. FWA-firefly search - FWA-FS) umesto procedure Gausove varnice koristi se FA mehanizam pretrage [164]. Algoritam testiran na problemima globalne optimizacije bez ograničenja i pokazano je da je diversifikacija smanjena, dok je, s druge strane, intenzifikacija značajno pojačana.

Slično kao i ABC, BA ne koristi eksplicitno operator ukrštanja, ali se zato koristi operator mutacije, čiji se intenzitet menja tokom izvršavanja algoritma. BA je generalno jako dobar algoritam, koji kombinuje prednosti FA i metaheuristika CS.

Međutim, ispitivanjem algoritma za globalne probleme bez ograničenja, uočeno je da pretraga koja se usmerava ka najboljim rešenjima u populaciji, u ranijim iteracijama, može da dovede do problema preuranjene konvergencije i algoritam pokazuje lošije prosečne rezultate. Takođe, uočeno je da je proces diversifikacije manje izražen u ranijim ciklusima.

Navedeni problemi mogu da se, prema rezultatima na proverenim instancama, isprave pojačanjem diversifikacije, ili primenom procedure koja pretragu ne usmerava ka trenutno najboljoj individui u populaciji. Predloženi hibrid, BA - mehanizam posmatrača (eng. BA-onlooker mechanism - BA-OM) kombinuje BA i strukturni element ABC algoritma [166]. Procedura pčela posmatrača koristi probabilističku selekciju na osnovu podobnosti individua, koja se dobro balansira sa selekcijom na osnovu najbolje individue metaheuristike BA.

BA-OM je predložen za benčmark probleme bez ograničenja i za praktičan višekriterijumski problem planiranja RFID mreže koji koristi pristup težinske sume za upravljanje ciljevima [166].

U ovom radu je dat širok pregled nedostataka metaheuristika rojeva i predloga njihovog ispravljanja pomoću hibridizacija. Algoritmi inteligencije rojeva ispitivani su s kritičkog aspekta, gde su utvrđeni njihovi nedostaci, i pokazano je da se utvrđeni nedostaci mogu ispraviti hibridizacijom. U toku istraživanja, implementirano je više osnovnih i hibridnih varijanti algoritama rojeva.

Predloženih sedam hibridnih algoritama rojeva, koji su prikazani i testirani, pokazali su se kao odlične metaheuristike koje postižu bolje performanse (testirani na istim instancama i pod istim uslovima) od vodećih algoritama inteligencije rojeva čiji su opisi i rezultati prikazani u vodećim međunarodnim časopisima. Osim opisa i implementacionih detalja, za svaki predloženi hibirdni algoritam dat je detaljan prikaz rezultata testiranja i komparativna analiza sa drugim vodećim algoritmima, na osnovu čega se vidi da hibridne metaheuristike postižu bolje rezultate, kako na

benčmark problemima globalne optimizacije, tako i na praktičnim problemima.

Na osnovu svega navedenog, u ovom radu je ostvareno:

- bolji uvid u efikasnost i kvalitet pojedinih algoritama rojeva na osnovu testiranja na širokom spektru optimizacionih problema;
- istraživanje međusobnih odnosa i komparativna analiza metaheuristika rojeva;
- detaljno ispitivanje i diskusija o prednostima i nedostacima jednih algoritma u odnosu na druge i
- unapređenje postojećih implementacija metaheuristika rojeva poboljšavanjem i nadogradnjom procesa pretrage putem hibridizacija (kvalitet ovih unapređenja verifikovan je upoređivanjem sa osnovnim algoritmima, kao i njihovim unapređenim verzijama iz literature).

Istraživanje opisano u ovom radu predstavlja doprinos u oblastima globalne optimizacije, prirodom inspirisanih algoritama i metaheuristika inteligencije rojeva.

Naučni doprinos ovog rukopisa verifikovan je publikovanjem više radova u svetskim časopisima i na međunarodnim konferencijama, od čega je četiri rada u časopisima sa SCI liste (tri iz kategorije M21 i jedan iz kategorije M23), 10 radova u drugim međunarodnim časopisima i 26 radova na međunarodnim konferencijama. Radovi su citirani 202 puta prema Google Scholar-u, 105 puta prema Scopus-u i 16 puta prema Thomson Reuters WoS-u.

Literatura

[1] X.-S. Yang, "Swarm intelligence based algorithms: a critical analysis," *Evolutionary Intelligence*, vol. 7, pp. 17–28, April 2014.

[2] F. Rothlauf, *Design of Modern Heuristics Principles and Application*. No. 1 in Natural Computing Series, Springer-Verlag Berlin Heidelberg, February 2011.

[3] J. Nocedal and S. Wright, *Numerical Optimization*. No. 2 in Springer Series in Operations Research and Financial Engineering, Springer-Verlag New York, 2006.

[4] F. Neumann and C. Witt, *Bioinspired Computation in Combinatorial Optimization*. Natural Computing Series, Springer-Verlag, November 2010.

[5] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[6] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Information Fusion*, vol. 191, pp. 1245–1287, January 2002.

[7] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, pp. 173–194, December 2011.

[8] T. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[9] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[10] K. Deb, "An efficient constraint-handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[11] T. Takahama, S. Sakai, and N. Iwane, "Constrained optimization by the constrained hybrid algorithm of particle swarm optimization and genetic algorithm," *Advances in Artificial Intelligence, LNCS*, vol. 3809, pp. 389–400, 2005.

[12] 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*, p. 24, 2006.

[13] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and Multidisciplinary Optimization*, vol. 41, pp. 853–862, June 2010.

[14] L. Ma, H. Chen, K. Hu, and Y. Zhu, "Hierarchical artificial bee colony algorithm for RFID network planning optimization," *The Scientific World Journal*, vol. 2014, no. Article ID 941532, p. 21, 2014.

[15] M. Caramia and P. Dell'Olmo, *Multi-objective Management in Freight Logistics*. No. 1 in ncreasing Capacity, Service Level and Safety with Optimization Algorithms, Springer-Verlag London, November 2008.

[16] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, vol. 1. Cambridge University Press, April 2009.

[17] C. S. Calude, E. Calude, and M. S. Queen, "Inductive complexity of p versus np problem," *Unconventional Computation and Natural Computation, LNCS*, vol. 7445, pp. 2–9, 2012.

[18] X.-S. Yang, "Swarm intelligence based algorithms: A critical analysis," *Evolutionary Intelligence*, vol. 7, pp. 17–28, April 2014.

[19] D. Ogan and M. Azizoglu, "A branch and bound method for the line balancing problem in u-shaped assembly lines with equipment requirements," *Journal of Manufacturing Systems*, vol. 36, pp. 46—-54, July 2015.

[20] R. Yokoyama, Y. Shinano, S. Taniguchi, M. Ohkura, and T. Wakui, "Optimization of energy supply systems by milp branch and bound method in consideration of hierarchical relationship between design and operation," *Energy Conversion and Management*, vol. 92, pp. 92—-104, March 2015.

[21] Y. Abdelsadek, F. Herrmann, I. Kacem, and B. Otjacques, "Branch-and-bound algorithm for the maximum triangle packing problem," *Computers & Industrial Engineering*, vol. 81, pp. 147–157, March 2015.

[22] S. Koziel and X.-S. Yang, *Computational Optimization, Methods and Algorithms*, vol. 356 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2011.

[23] R. Martini and G. Reinelt, *The Linear Ordering Problem Exact and Heuristic Methods in Combinatorial Optimization*, vol. 175 of *Applied Mathematical Sciences*. Springer-Verlag Berlin Heidelberg, 2011.

[24] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. No. 1, Wiley Publishing, July 2009.

[25] K. E. Train and W. W. Wilson, "Monte carlo analysis of sp-off-rp data," *Journal of Choice Modelling*, vol. 2, no. 1, pp. 101–117, 2009.

[26] C. Matthews, L. Wright, and X.-S. Yang, "Sensitivity analysis, optimization, and sampling methodds applied to continous models," *National Physical Laboratory Report*, 2009.

[27] D. Jungnickel, *Graphs, Networks and Algorithms*, vol. 5 of *Algorithms and Computation in Mathematics*. Springer-Verlag Berlin Heidelberg, 2013.

[28] F. Glover and C. McMillan, "The general employee scheduling problem: an integration of ms and ai," *Computers and Operations Research, Special issue: Applications of integer programming*, vol. 13, no. 5, pp. 563–573, 1986.

[29] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 327–333, December 1997.

[30] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*. No. 1 in Natural Computing Series, Springer-Verlag Berlin Heidelberg, 2005.

[31] I. Boussaid, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82—-117, July 2013.

[32] N. Mladenovic and P. Hansen, "Variable neighborhood search," *Journal of Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[33] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decision Sciences*, vol. 8, pp. 156–166, January 1977.

[34] C. Darvin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, First Edition*. London: John Murray, 1859.

[35] K. Kirkpatric, C. D. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science Magazine*, vol. 220, no. 4598, pp. 671–680, 1983.

[36] X.-S. Yang, *Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics*. Cambridge International Science Publishing, January 2008.

[37] I. K. Panigrahi, Y. Shi, and M.-H. Lim, *Handbook of Swarm Intelligence*, vol. 8 of *Adaptation, Learning, and Optimization*. Springer-Verlag, 2011.

[38] D. W. Corne, A. Reynolds, and E. Bonabeau, *Swarm Intelligence*. Handbook of Natural Computing, Springer-Verlag, 2012.

[39] D. Marco and M. G. Luca, "Ant colonies for the travelling salesman problem," *Biosystems*, vol. 43, pp. 73–81, July 1997.

[40] D. M., M. V., and C. A., "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[41] J. Kennedy and R. Eberhart, "Particle swarm optimization," vol. 4, pp. 1942–1948, 1995.

[42] E. R. C. and K. J., "A new optimizer using particle swarm theory," in *Proceedings of the 6th international symposium on micromachine and human science*, pp. 39–43, IEEE, 1995.

[43] X.-S. Yang and S. Deb, "Cuckoo search via levy flights," in *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, pp. 210–214, 2009.

[44] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms: Second Edition*. Luniver Press, July 2010.

[45] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, pp. 17–35, January 2013.

[46] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun Nonlinear Sci Numer Simulat*, vol. 17, pp. 4831—-4845, 2012.

[47] C. Vincenzo and G. Nicosia, "Multiple learning using immune algorithms," in *Proceedings of 4th International Conference on Recent Advances in Soft Computing*, pp. 102–107, 2002.

[48] L. Castro and F. J. V. Zuben, "The status of the p versus np problem," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 239–251, June 2002.

[49] C. A. C. Coello and N. C. Cortés, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, pp. 163–190, June 2005.

[50] N. Bacanin and M. Tuba, "Artificial immune system algorithm framework for bound-constraint numerical problems," in *Proceedings of the 12th International Conference on Artificial Intelligence*, 2013.

[51] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," *Technical Report - TR06*, pp. 1–10, 2005.

[52] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," *Foundations of Fuzzy Logic and Soft Computing, LNCS*, vol. 4529, pp. 789–798, 2007.

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

[54] X.-S. Yang, "Firefly algorithms for multimodal optimization," *Stochastic Algorithms: Foundations and Applications, LNCS*, vol. 5792, pp. 169–178, 2009.

[55] C. Dai, W. Chen, and Y. Zhu, "Seeker optimization algorithm," vol. „ pp. 225–229, 2006.

[56] C. Dai, Y. Zhu, and W. Chen, "Seeker optimization algorithm," *Lecture Notes in Computer Science*, vol. 4456, pp. 167–176, 2007.

[57] C. Dai, W. Chen, L. Ran, Y. Zhang, and Y. Du, "Human group optimizer with local search," *Proceedings of the Second international conference on Advances in swarm intelligence ICSI'11*, vol. Part I, pp. 310–320, 2011.

[58] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," *Studies in Computational Intelligence*, vol. 284, pp. 65—74, November 2010.

[59] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," *Advances in Swarm Intelligence, LNCS*, vol. 6145, pp. 355–364, June 2010.

[60] I. Tsoulos and A. Stavrakoudisb, "Enhancing pso methods for global optimization," *Applied Mathematics and Computation*, vol. 216, pp. 2988—-3001, July 2010.

[61] L. Zhang, Y. Tang, C. Hua, and X. Guan, "A new particle swarm optimization algorithm with adaptive inertia weight based on bayesian techniques," *Applied Soft Computing*, vol. 28, pp. 138–149, June 2015.

[62] N. Bacanin, "Implementation and performance of an object-oriented software system for cuckoo search algorithm," *International Journal of Mathematics and Computers in Simulation*, vol. 6, pp. 185–193, December 2010.

[63] 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*, vol. 21, no. 2, pp. 300–311, 2010.

[64] S. Zheng, A. Janecek, J. Li, and Y. Tan, "Dynamic search in fireworks algorithm," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC 2014)*, pp. 3222–3229, July 2014.

[65] K. Ding, S. Zheng, and Y. Tan, "A gpu-based parallel fireworks algorithm for optimization," in *Proc. of the 15th annual conference on Genetic and evolutionary computation (GECCO '13)*, pp. 9–16, 2013.

[66] Darvis and B. Akay, "A modified artificial bee colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing*, vol. 11, no. 3, pp. 3021–3031, 2011.

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

[68] M. Tuba, N. Bacanin, and N. Stanarevic, "Adjusted artificial bee colony (ABC) algorithm for engineering problems," *WSEAS Transactions on Computers*, vol. 11, no. 4, pp. 111–120, 2012.

[69] G. Zhua and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied mathematics and computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[70] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[71] S. Łukasik and S. Żak, "Firefly algorithm for continuous constrained optimization tasks," *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems, LNCS*, vol. 5796, pp. 97–106, 2009.

[72] X.-S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Engineering with Computers*, vol. 29, pp. 175–184, January 2012.

[73] X.-S. Yang and S. Deb, "Engineering optimization by cuckoo search," *International Journal of Mathematical Modeling and Numerical Optimization*, vol. 1, no. 4, pp. 330–343, 2010.

[74] X. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Engineering Computations*, vol. 29, pp. 464––483, January 2012.

[75] X. Yang, "Bat algorithm for multi-objective optimization," *International Journal of Bio-Inspired Computation*, vol. 3, pp. 267—-274, December 2011.

[76] H. Zhu, Y. Wang, K. Wang, and Y. Chen, "Modified particle swarm optimization for nonconvex economic dispatch problems," *International Journal of Electrical Power & Energy Systems*, vol. 69, pp. 304—312, July 2015.

[77] H. Zhu, Y. Wang, K. Wang, and Y. Chen, "Particle swarm optimization (PSO) for the constrained portfolio optimization problem," *Expert Systems with Applications*, vol. 38, no. 1, pp. 10161—-10169, 2011.

[78] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (abc) algorithm and applications," *Artificial Intelligence Review*, vol. 42, pp. 21–57, June 2014.

[79] B. S. P. Mishra, S. Dehuri, and G.-N. Wang, "A state-of-the-art review of artificial bee colony in the optimization of single and multiple criteria," *Inter-*

national Journal of Applied Metaheuristic Computing, vol. 4, no. 4, pp. 23–45, 2013.

[80] R. Akbari, V. Zeighami, and I. Akbari, "An abc-genetic method to solve resource constrained project scheduling problem," *Artificial Intelligence Research*, vol. 1, pp. 185–197, December 2012.

[81] I. Brajevic and M. Tuba, "Cuckoo search and firefly algorithm applied to multilevel image thresholding," in *Cuckoo Search and Firefly Algorithm: Theory and Applications* (X.-S. Yang, ed.), vol. 516 of *Studies in Computational Intelligence*, pp. 115–139, Springer International Publishing, 2014.

[82] A. H. Gandomi, X.-S. Yang, and A. H. Alavic, "Mixed variable structural optimization using firefly algorithm," *Computers & Structures*, vol. 89, no. 23–24, pp. 2325—-2336, 2011.

[83] M. Tuba, N. Bacanin, and B. Pelevic, "Framework for constrained portfolio selection by the firefly algorithm," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 7, no. 10, pp. 1888–896, 2014.

[84] P. Dasgupta and S. Das, "A discrete inter-species cuckoo search for flowshop scheduling problems," *Computers & Operations Research*, vol. 60, pp. 111—-120, August 2015.

[85] M. Kumar and T. K. Rawat, "Optimal design of fir fractional order differentiator using cuckoo search algorithm," *Expert Systems with Applications*, vol. 42, pp. 3433—-3449, May 2015.

[86] C. Dai, W. Chen, Y. Zhu, and X. Zhang, "Reactive power dispatch considering voltage stability with seeker optimization algorithm," *Electric Power Systems Research*, vol. 79, no. 10, pp. 1462–1471, 2009.

[87] K. Krishnanand, P. Rout, B. Panigrahi, and A. Mohapatra, "Solution to non-convex electric power dispatch problem using seeker optimization algorithm," *LNCS Swarm, Evolutionary, and Memetic Computing*, vol. 6466, pp. 537–544, 2010.

[88] B. Abhik, V. Mukherjee, and S. Ghoshal, "Seeker optimization algorithm for load-tracking performance of an autonomous power system," *International Journal of Electrical Power & Energy Systems*, vol. 43, no. 1, pp. 1162–1170, 2012.

[89] C. Dai, W. Chen, and Y. F. Zhu, "Seeker optimization algorithm for digital IIR filter design," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 5, pp. 1710–1718, 2010.

[90] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "An effective krill herd algorithm with migration operator in biogeography-based optimization," *Applied Mathematical Modelling*, vol. 38, pp. 2454—-2462, May 2014.

[91] N. Bacanin, M. Tuba, and B. Pelevic, "Krill herd (KH) algorithm for portfolio optimization," in *14th International Conference on Mathematics and Computers in Business and Economics (MCBE '13)*, pp. 39–44, 2013.

[92] N. Bacanin and M. Tuba, "Fireworks algorithm applied to constrained portfolio optimization problem," in *Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC 2015)*, pp. 1242–1249, May 2015.

[93] T. Ting, X.-S. Yang, S. Cheng, and K. Huang, "Hybrid metaheuristic algorithms: Past, present, and future," *Recent Advances in Swarm Intelligence and Evolutionary Computation Studies in Computational Intelligence*, vol. 585, pp. 71–83, December 2015.

[94] C. I. and E. Kyriakides, "Hybrid ant colony-genetic algorithm (gaapi) for global continuous optimization," *IEEE Transcations on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, pp. 234—-245, February 2012.

[95] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proceedings of IEEE Congress on Evolutionary Computation 2006 CEC 2006*, pp. 207–214, 2006.

[96] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms," *Technical Report C3P 826*, 1989.

[97] F. Neri and C. Cotta, "Hybrid taguchi-genetic algorithm for global numerical optimization," *Swarm and Evolutionary Computation*, vol. 2, pp. 1—-14, November 2011.

[98] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation—past, present & future [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 2, pp. 24–31, May 2010.

[99] F. Neri and E. Mininno, "Memetic compact differential evolution for cartesian robot control," *IEEE Computational Intelligence Magazine*, vol. 5, pp. 54–65, May 2010.

[100] F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms*, vol. 379 of *Studies in Computational Intelligence*. Springer-Verlag, 2012.

[101] H. Lourenço, O. Martin, and T. Stützle, "Iterated local search," *Handbook of Metaheuristics*, vol. 57, pp. 320–353, 2003.

[102] A. Caponio, G. Cascella, F. Neri, and N. Salvatore, "A fast adaptive memetic algorithm for online and offline control design of pmsm drives," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 28–41, 2007.

[103] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi, "A direct first principles study on the structure and electronic properties of bexzn1-xo," *Applied Physics Letters*, vol. 16, no. 4, pp. 529–555, 2008.

[104] J. Hernández Mejía, O. Schütze, and K. Deb, "A memetic variant of r-nsga-ii for reference point problems," vol. 288, pp. 247–260, 2014.

[105] Z. Wu, X. Xia, and J. Zhang, "MMODE: A memetic multiobjective differential evolution algorithm," vol. 7928, pp. 422–430, 2013.

[106] L. Feng, A.-H. Tan, M.-H. Lim, and S. Jiang, "Band selection for hyperspectral images using probabilistic memetic algorithm," *Soft Computing*, pp. 1–9, 2014.

[107] A. de Freitas, F. Guimarães, R. Pedrosa Silva, and M. Souza, "Memetic self-adaptive evolution strategies applied to the maximum diversity problem," *Optimization Letters*, vol. 8, no. 2, pp. 705–714, 2014.

[108] L. Wang and J. Liu, "A scale-free based memetic algorithm for resource-constrained project scheduling problems," vol. 8206, pp. 202–209, 2013.

[109] A. Lančinskas and J. Žilinskas, "Parallel multi-objective memetic algorithm for competitive facility location," pp. 354–363, 2014.

[110] F. Rodriguez, C. Garcia-Martinez, and M. Lozano, "Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test," *IEEE Transactions on Evolutionary Computation*, vol. 16, pp. 787–800, December 2012.

[111] C. Grosan and A. Abrahaml, "Hybrid evolutionary algorithms: Methodologies, architectures, and reviews," *Hybrid Evolutionary Algorithms Studies in Computational Intelligence*, vol. 75, pp. 1–17, 2007.

[112] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of Operional Research Society*, vol. 64, pp. 1695—-1724, July 2013.

[113] X.-S. Yang, M. Karamanoglu, T. O. Ting, and Y.-X. Zhao, "Applications and analysis of bio-inspired eagle strategy for engineering optimization," *Neural Computing and Applications*, vol. 25, pp. 411–420, August 2014.

[114] X.-S. Yang, T. O. Ting, and M. Karamanoglu, "Random walks, lévy flights, markov chains and metaheuristic optimization," *Future Information Communication Technology and Applications Lecture Notes in Electrical Engineering*, vol. 235, pp. 1055–1064, May 2013.

[115] T. O. Ting, K. P. Wong, and C. Y. Chung, "Advances in machine learning and cybernetics lecture notes in computer science," *Future Information Communication Technology and Applications Lecture Notes in Electrical Engineering*, vol. 3930, pp. 908–917, August 2006.

[116] W.-J. Zhang and X.-F. Xie, "DEPSO: hybrid particle swarm with differential evolution operator," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3816–3821, October 2003.

[117] X.-F. Xie, W.-J. Zhang, and Z.-L. Yang, "A dissipative particle swarm optimization," in *Proceedings of the Fourth Congress on Evolutionary Computation (CEC 2002)*, vol. 2, pp. 1456—-1461, 2002.

[118] E.-G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of Heuristics*, vol. 8, pp. 541–564, September 2002.

[119] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 3rd Edition.* Wiley-IEEE Press, February 2006.

[120] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[121] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing, 1989.

[122] F. Rodriguez, C. Garcia-Martinez, and M. Lozano, "Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: Taxonomy, comparison, and synergy test," *IEEE Transactions on Evolutionary Computation*, vol. 16, pp. 787–800, December 2012.

[123] K. C. Tan, Q. Yu, C. Heng, and T. Lee, "Evolutionary computing for knowledge discovery in medical diagnosis," *Artificial Intelligence in Medicine*, vol. 27, pp. 129—-154, May 2003.

[124] M. A. Zmuda, M. M. Rizki, and L. A. Tamburinoc, "Hybrid evolutionary learning for synthesizing multi-class pattern recognition systems," *Applied Soft Computing*, vol. 2, pp. 269–282, February 2003.

[125] L. Wang, "A hybrid genetic algorithm–neural network strategy for simulation optimization," *Applied Mathematics and Computation*, vol. 170, pp. 1329––1343, November 2005.

[126] F. Herrera and M. Lozano, "Adaptive control of the mutation probability by fuzzy logic controllers," *Parallel Problem Solving from Nature PPSN VI Lecture Notes in Computer Science*, vol. 1917, pp. 335–344, July 2000.

[127] S. Li, M. Tan, I. W. Tsang, and J. T. yau Kwok, "A hybrid pso-bfgs strategy for global optimization of multimodal functions," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 41, no. 4, pp. 1003–1014, 2011.

[128] J.-T. Tsai, T.-K. Liu, and J.-H. Chou, "Hybrid taguchi-genetic algorithm for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 365—-377, August 2004.

[129] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid taguchi-genetic algorithm for global numerical optimization," *IEEE Transactions on Pattern Analysis and Machine*, vol. 26, pp. 1424–1437, November 2004.

[130] N. Bacanin and M. Tuba, "Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators," *Studies in Informatics and Control*, vol. 21, pp. 137–146, June 2012.

[131] E. Mezura-Montes (editor), *Constraint-Handling in Evolutionary Optimization*, vol. 198 of *Studies in Computational Intelligence*. Springer-Verlag, 2009.

[132] E. Mezura-Montes and C. A. Coello-Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[133] E. Mezura-Montes, M. Damian-Araoz, and O. Cetina-Domingez, "Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization," in *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, July 2010.

[134] B. G. Tessema and G. G. Yen, "A self-adaptive penalty function based algorithm for constrained optimization," *IEEE Congress on Evolutionary Computation 2006 (CEC'2006)*, pp. 246–253, 2006.

[135] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, pp. 80–92, January 2008.

[136] N. Bacanin and M. Tuba, "Multi-objective RFID network planning by artificial bee colony algorithm with genetic operators," vol. 9140, pp. 247–254, Springer International Publishing, June 2015.

[137] N. Bacanin, M. Tuba, and I. Strumberger, "RFID network planning by abc algorithm hybridized with heuristic for initial number and locations of readers," in *Proceedings of the 17th UKSIM-AMSS International Conference on Modeling and Simulation*, pp. 39–44, March 2015.

[138] L. Ma, K. Hu, Y. Zhu, and H. Chen, "Cooperative artificial bee colony algorithm for multi-objective RFID network planning," *Journal of Network and Computer Applications*, vol. 42, pp. 143–162, June 2014.

[139] Q. Guan, Y. Liu, Y. Yang, and W. Yu, "Genetic approach for network planning in the RFID systems," in *Proc. of the Sixth International Conference on Intelligent Systems Design and Applications*, vol. 2, pp. 567–572, IEEE, 2006.

[140] Y. J. Gong, M. Shen, J. Zhang, O. Kaynak, W. N. Chen, and Z. H. Zhan, "Optimizing RFID network planning by using a particle swarm optimization algorithm with redundant reader elimination," *IEEE Transactions on Industrial Informatics*, vol. 8, pp. 900–912, November 2012.

[141] H. Chen, Y. Zhu, K. Hu, and T. Ku, "RFID network planning using a multi-swarm optimizer," *Journal of Network and Computer Applications*, vol. 34, pp. 888–901, May 2011.

[142] H. Chen, Y. Zhu, and K. Hu, "Multi-colony bacteria foraging optimization with cell-to-cell communication for RFID network planning," *Applied Soft Computing*, vol. 10, pp. 539–547, March 2010.

[143] S. Lu and S. Yu, "A fuzzy k-coverage approach for RFID network planning using plant growth simulation algorithm," *Journal of Network and Computer Applications*, vol. 39, pp. 280–291, March 2014.

[144] M. Tuba and N. Bacanin, "Artificial bee colony algorithm hybridized with firefly metaheuristic for cardinality constrained mean-variance portfolio problem," *Applied Mathematics & Information Sciences*, vol. 8, pp. 2831–2844, November 2014.

[145] X.-S. Yang, "Firefly algorithm, lévy flights and global optimization," *Research and Development in Intelligent Systems*, vol. XXVI, pp. 209–218, 2010.

[146] M. Tuba and N. Bacanin, "Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems," *Neurocomputing*, vol. 143, pp. 197–207, November 2014.

[147] N. Bacanin and M. Tuba, "Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint," *The Scientific World Journal, special issue Computational Intelligence and Metaheuristic Algorithms with Applications,*, vol. 2014, no. Article ID 721521, p. 16, 2014.

[148] I. Usta and Y. M. Kantar, "Mean-variance-skewness-entropy measures: A multi-objective approach for portfolio selection," *Entropy*, vol. 13, no. 1, pp. 117–133, 2011.

[149] J. Xu, X. Zhou, and D. D. Wu, "Portfolio selection using λ-mean and hybrid entropy," *Annals of Operations Research*, vol. 185, no. 1, pp. 213–229, 2011.

[150] H. R. Golmakani and M. Fazel, "Constrained portfolio selection using particle swarm optimization," *Expert Systems with Applications*, vol. 38, no. 1, pp. 8327—-8335, 2011.

[151] X. Huang, "An entropy method for diversified fuzzy portfolio selection," *International Journal of Fuzzy Systems*, vol. 14, no. 1, pp. 160–165, 2012.

[152] T. Cura, "Particle swarm optimization approach to portfolio optimization," *Nonlinear Analysis: RealWorld Applications*, vol. 10, no. 4, pp. 2396–2406, 2008.

[153] K. Deb, "Optimization for engineering design, algorithms and examples," *Computer Methods in Applied Mechanics and Engineering*, p. 396, 2005.

[154] T. Chang, N. Meade, J. Beasley, and Y. Sharaiha, "Heuristics for cardinality constrained portfolio optimization," *Computers and Operations Research*, vol. 27, no. 13, pp. 1271—-1302, 2000.

[155] Z. Wang, S. Liu, and X. Kong, "Artificial bee colony algorithm for portfolio optimization problems," *International Journal of Advancements in Computing Technology*, vol. 4, no. 4, pp. 8–16, 2012.

[156] Z. Wang, R. Ouyang, and X. Kong, "A hybrid artificial bee colony for portfolio optimization," *Journal of Theoretical and Applied Information Technologyh*, vol. 49, no. 1, pp. 94–100, 2013.

[157] M. Tuba and I. Brajevic, "Modified seeker optimization algorithm for image segmentation by multilevel thresholding," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 7, no. 3, pp. 867–875, 2013.

[158] E. Mezura-Montes, M. Damian-Araoz, and O. Cetina-Domingez, "Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization," *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[159] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, 2008.

[160] T. P. Runarsson and X. Yao, "Search biases in constrained evolutionary optimization," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 35, no. 2, pp. 233–243, 2005.

[161] S. B. Hamida and M. Schoenauer, "ASCHEA: new results using adaptive segregational constraint handling," *Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 884–889, 2002.

[162] E. Mezura-Montes and A. C. Coello Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, 2005.

[163] A. E. M. Zavala, A. H. Aguirre, and E. R. V. Diharce, "Constrained optimization via particle evolutionary swarm algorithm (PESO)," *Proceedings of the conference on Genetic and evolutionary computation GECCO*, pp. 209–216, 2005.

[164] N. Bacanin, M. Tuba, and M. Beko, "Hybridized fireworks algorithm for global optimization," in *Mathematical Methods and Systems in Science and Engineering*, pp. 108–114, 2015.

[165] Y. Tan and Z. Xiao, "Clonal particle swarm optimization and its applications," in *Proc. of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 2303–2309, 2007.

[166] M. Tuba and N. Bacanin, "Hybridized bat algorithm for multi-objective radio frequency identification (RFID) network planning," in *Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC 2015)*, pp. 499–506, May 2015.

[167] Y. Shi and R. C. Eberchar, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1945–1950, IEEE, 1999.

[168] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1671–1676, IEEE, 2002.

[169] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler ,maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 204–210, December 2004.

[170] K. Parsopoulos and M. Vrahatis, "UPSO :a unified particle swarm optimization scheme," *Lecture Notes in Computer Science*, pp. 868–873, 2004.

[171] K. Veeramachaneni, T. Peram, C. Mohan, and L. A. Osadciw, "Optimization using particle swarms with near neighbor interactions," in *Lecture Notes Computer Science*, pp. 110–121, Springer Verlag, 2003.

[172] B. Niu, Y. Zhu, X. He, and H. Wu, "MCPSO: a multi-swarm cooperative particle swarm optimizer," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1050–1062, 2007.

[173] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, no. 214, pp. 108–132, 2009.

Biografija

Nebojša Bačanin Džakula rođen je 13.01.1983. godine u Beogradu. Osnovnu školu i gimnaziju završio je u Beogradu. Osnovne studije završio je na Fakultetu za poslovne studije Megatrend univerziteta u Beogradu sa prosečnom ocenom 9,38. Master studije na Fakultetu organizacionih nauka Univerziteta u Beogradu, smer Informacioni sistemi, završio je 2008. godine sa prosečnom ocenom 10,00. Na doktorskim studijama na Matematičkom fakultetu Univerziteta u Beogradu, smer informatika, položio je sve ispite predviđene nastavnim planom i programom sa prosečnom ocenom 10.00.

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

Angažovan je na projektu III-44006 iz programa integralnih interdiscipliranih istraživanja koji se finansira od strane Ministarstva za obrazovanje, nauku i tehnološki razvoj Republike Srbije u periodu 01.01.2011 – 31.12.2015, pod nazivom „Razvoj novih informaciono-komunikacionih tehnologija, korišćenjem naprednih matematičkih metoda, sa primenama u medicini, energetici, telekomunikacijama, e-upravi i zaštiti nacionalne baštine“, čiji je rukovodilac dr Zoran Ognjanović, Matematički institut Srpske Akademije nauka i umetnosti.