PRILAGOĐAVANJE ALGORITAMA INTELIGENCIJE ROJEVA ZA RAZLIČITE PROSTORE PRETRAGE

By Romana Capor Hrošik

WORD COUNT 41755 TIME SUBMITTED 15-JUL-2020 01:08PM
PAPER ID 61206999

Univerzitet Singidunum

Departman za poslediplomske studije

Doktorska disertacija

PRILAGOĐAVANJE ALGORITAMA INTELIGENCIJE ROJEVA ZA RAZLIČITE PROSTORE PRETRAGE

Mentor:
prof. dr. Milan Tuba

Kandidat:
Romana Capor Hrošik

Beograd, 2020.

Singidunum University

Graduate school

PhD Thesis

SWARM INTELLIGENCE ALGORITHMS ADAPTATION FOR VARIOUS SEARCH SPACES

Mentor:

prof. Milan Tuba, PhD

Candidate:

Romana Capor Hrošik

Belgrade, 2020.

* * * * * * * * *

Zahvaljujem se svojem mentoru, prof. dr. Milanu Tubi koji me je uveo u problematiku znanstvenog istraživanja te omogućio nastavak mojeg znanstvenog rada. Posebnu zahvalnost dugujem svojoj obitelji koja je razlog da sam danas tu gdje jesam.

Prilagođavanje algoritama inteligencije rojeva za različite prostore pretrage

Sažetak – U današnje vrijeme postoji mnogo algoritama inteligencije rojeva koji se uspiješno koriste za rešavanje raznih teških problema optimizacije. Zajednički elementi svih ovih algoritama su operator za lokalnu pretragu (eksploataciju) oko pronađenih obećavajućih rješenja i operator globalne pretrage (eksploracije) koji pomaže u bijegu iz lokalnih optimuma. Algoritmi inteligencije rojeva obično se inicijalno testiraju na neograničenim, ograničenim ili visoko-dimenzionalnim skupovima standardnih test funkcija. Nadalje, mogu se poboljšati, prilagoditi, izmijeniti, hibridizirati, kombinirati s lokalnom pretragom. Konačna svrha je korištenje takve metaheuristike za optimizaciju problema iz stvarnog svijeta. Domeni rješenja odnosno prostori pretrage praktičnih teških problema optimizacije mogu biti različiti. Rješenja mogu biti vektori iz skupa realnih brojeva, cijelih brojeva ali mogu biti i kompleksnije strukture. Algoritmi inteligencije rojeva moraju se prilagoditi za različite prostore pretrage što može biti jednostavno podešavanje parametera algoritma ili prilagodba za cjelobrojna rješenja jednostavnim zaokruživanjem dobivenih realnih rješenja ali za pojedine prostore pretrage potrebnao je skoro kompletno prepravljanja algoritma uključujući i operatore ekploatacije i ekploracije zadržavajući samo proces vođenja odnosno inteligenciju roja.

U disertaciji je predstavljeno nekoliko algoritama inteligencije rojeva i njihova prilagodba za različite prostore pretrage i primjena na praktične probleme. Ova disertacija ima za cilj analizirati i prilagoditi, u zavisnosti od funkcije cilja i prostora rješenja, algoritme inteligencije rojeva. Predmet disertacije uključuje sveobuhvatan pregled postojećih implementacija algoritama inteligencije rojeva. Disertacija također obuhvaća komparativnu analizu, prikaz slabosti i snaga jednih algoritama u odnosu na druge zajedno s istraživanjem prilagodbi algoritama inteligencije rojeva za različite prostore pretrage i njihova primjena na praktične problem. Razmatrani su problemi sa realnim rješenjima kao što su optimizacija stroja potpornih vektora, grupiranje podataka, sa cijelobrojnim rješenjima kao što je slučaj problema segmentacije digitalnih slika i za probleme gdje su rješenja posebne strukture kao što su problemi planiranja putanje robota i triangulacije minimalne težine.

Modificirani i prilagođeni algoritmi inteligencije rojeva za različite prostore pretrage i primjenih na praktične probleme testirani su na standardnim skupovima test podataka i uspoređeni s drugim suvremenim metodama za rješavanje promatranih problema iz literature. Pokazane su uspješne prilagodbe algoritama inteligencije rojeva za razne prostore pretrage. Ovako prilagođeni algoritmi su u svim slučajevima postigli bolje rezultate u usporedbi sa metodama iz literature, što dovodi do zaključka da je moguće prilagoditi algoritme inteligencije rojeva za razne prostore pretrage uključujući i kompleksne strukture i postići bolje rezultate u usporedbi sa metodama iz literature.

Ključne reči: optimizacija, NP-teški optimizacijski problemi, algoritmi inteligencije rojeva, prilagodjavanje algoritama, prostor pretrage, praktične primjene.

Adjustment of swarm intelligence algorithms for different search spaces

Abstract – Nowadays, there are many swarm intelligence algorithms that are successfully used to solve various hard optimization problems. Common elements of all these algorithms are the operator for local search (exploitation) around the promising solutions that have been found and the global search (exploration) operator that helps to escape from local optima. Swarm intelligence algorithms are usually initially tested on unconstrained, constrained, or high-dimensional sets of standard test functions. Furthermore, they can be improved, adapted, modified, hybridized, combined with local search. The ultimate purpose is to use such metaheuristics to real-world optimization problems. The domains of solutions or search spaces for practical hard optimization problems can be different. Solutions can be vectors of real or integers numbers, however they can also be more complex structures. Swarm intelligence algorithms have to be adapted for different search spaces, which can be a simple adoptation of algorithm parameters or adaptation for integer solutions by simply rounding the obtained real solutions, but for some search spaces, the algorithm needs to be almost completely redesigned, including exploitation and exploration operators keeping only the guiding process or swarm intelligence.

Several swarm intelligence algorithms and their adaptations for different search spaces and application to practical problems are presented in this dissertation Goals of dissertation are to analyze and adapt swarm intelligence algorithms depending on the fitness function and search space. The topic of this dissertation includes a comprehensive review of existing implementations of swarm intelligence algorithms. The dissertation also includes comparative analysis, presentation of weaknesses and strengths of some algorithms in relation to others together with research of adaptation of swarm intelligence algorithms for different search spaces and their application to practical problems. Problems with real solutions such as support vector machine optimization, clustering, with integer solutions such as digital image segmentation problems and problems where solutions have a special structure such as robot path planning and minimum weight triangulation are considered.

Modified and adapted swarm intelligence algorithms for different search spaces and

applied to practical problems were tested on standard test data sets and compared with other modern methods from the literature for solving the considered problems. Successful adaptations of swarm intelligence algorithms for various search spaces are shown. Such adapted algorithms achieved better results for all cases compared to the methods from the literature, which leads to the conclusion that it is possible to adapted swarm intelligence algorithms for various search spaces including complex structures and achieve better results compared to methods from the literature.

Keywords: optimization, NP-hard optimization problems, swarm intelligence algorithms, algorithm adaptation, search spaces, practical applications.

Popis slika

1 Minimum od f(x) je jednak maksimumu od -f(x) 19
2 Metode optimizacije 32
3 Optimizacijski algoritmi 34
4 Primjer optimalne hiperravnine i margine 66
5 Primjer optimalne hiperravnine i margine za različite vrijednosti parametra C (C=1 lijevo, C=10 sredina i C=1000 desno) 67
6 Primjer linearno nerazdvojivog skupa podataka 68
7 Primjer granice dobivene za različite vrijednosti parametra γ (γ=10 lijevo, γ=0.01 sredina i γ=0.00001 desno) 69
8 Primjer rada algoritma k-sredina 72
9 Primjer usporedbe intuitivnog grupiranja i aktualnog grupiranja s algoritmom k-sredina 73
10 Model putanje 84
11 Primjer najbolje putanje leta kada je dimenzija problema jednaka 10 (d=10) 87
12 Primjeri različitih triangulacija istig skupa točaka 89
13 Konvergencija predloženog algoritma primijenjenog na skup podataka *Mouse* 105
14 Konvergencija predloženog algoritma primijenjenog na skup podataka *Robot* 106
15 Grupiranje skupa podataka *Mouse* s (a) algoritmom k-sredina i (b) predloženom metodom u ovom radu 107
16 Grupiranje skupa podataka *Robot* s (a) algoritmom k-sredina i (b) predloženom metodom u ovom radu 108
17 Originalne slike i njihove segmentirane slike dobivene našom predloženom metodom KM-FA algoritmom 118

Popis tabela

1 Najčešće korištene jezgrene funkcije 68
2 Distribucija instanci u skupu podataka 96
3 Rezultati usporedbe za različite omjere trening i test skupa 98
4 Informacije o korištenim skupovima podataka 109
5 Usporedba podataka za skup podataka *Page Blocks* 110
6 Usporedba podataka za skup podataka *Spambase* 110
7 Inicijalni parametri za FA 115
8 Usporedba između KM-FA algoritma i pristupa od Nanda i suradnika 116
9 Informacije o poznatim područjima prijetnje 123
10 Komparacija dobivenih rezultata između osam algoritama preko 100 nezavisnih ciklusa za d=20 124
11 Usporedba rezultata optimizacije povezanih sa statističkim parametrima za osam algoritama za različite vrijednosti dimenzije d preko 100 ciklusa i 6000 procjena funkcije 125
12 Optimale težine i vrijeme računanja pronađene iscrpnim pretraživanjem i AABC algoritmom 130
13 Usporedba srednjih vrijednosti i standardnih odstupanja dobivenih za SA, PSO i AABC za pet slučajnih slučajeva tijekom 50 ciklusa. . . . 131
14 Usporedba najgorih, srednjih, najboljih vrijednosti i standardnih odstupanja dobivenih u 50 ciklusa za SA, PSO i AABC za sedam slučajno generiranih slučajeva. 133

Popis algoritama

1 Predložak za ACO algoritam 45
2 ABC Algoritam 47
3 Algoritam krijesnice 50
4 Pseudokod algoritma šišmiša 53
5 Pseudokod brain storm optimizacijskog algoritma 56
6 Pseudo kod EHO algoritma 58
7 BBFWA algoritam [67] 62
8 Pseudokod predloženog algoritma za grupiranje podataka web inteligencije 104
9 Pseudokod za predloženi KM-FA algoritam 113

Sadržaj

Popis slika **6**

Popis tabela **7**

1 Uvod **11**

2 Optimizacija **15**

2.1 Klasifikacija problema optimizacije 18

2.2 Optimizacija s jednom ili više funkcija cilja 23

2.3 Optimizacija bez ograničenja 25

2.4 Optimizacija s ograničenjima 26

2.5 Teški optimizacijski problemi 28

3 Algoritmi za rješavanje problema optimizacije **31**

3.1 Klasifikacija metoda optimizacije 32

4 Algoritmi inteligencije rojeva **37**

4.1 Osnove u algoritmima inteligencije rojeva 39

4.2 Značajni predstavnici algoritama inteligencije rojeva 44

4.2.1 Algoritam optimizacije kolonijom mrava 44

4.2.2 Algoritam optimizacije rojem čestica 46

4.2.3 Algoritam umjetne kolonije pčela 47

4.2.4 Algoritam krijesnica 49

4.2.5 Algoritam šišmiša 51

4.2.6 Brain storm optimizacijski algoritam 53

4.2.7 Algoritam krdo slonova 57

4.2.8 Algoritam vatrometa 59

5 Prilagođavanje algoritama inteligencije rojeva za različite prostore pretrage **63**

5.1 Prostor pretrage za realna rješenja 63

5.1.1 Optimizacija stroja potpornih vektora 64

5.1.2 Primjena algoritma k-sredina na praktične probleme 71

5.2 Prostor pretrage za cjelobrojna rješenja 74
5.2.1 Segmentacija digitalnih slika 76
5.3 Prostor pretrage za složene strukture 79
5.3.1 Problem planiranja putanje 80
5.3.2 Trinagulacija minimalne težine 87

6 Rezultati istraživanja prilagođavanja algoritama inteligencije rojeva za praktične probleme sa različitim prostorima pretrage 92
6.1 Rezultati prilagodjavanja algoritama za prostor realnih rješenja . . . 93
6.1.1 Optimizacija stroja potpornih vektora za otkrivanje eritematoloških–skvamoznih bolesti algoritmom krda slonova 93
6.1.2 Grupiranje podataka web inteligencije ogoljenim algoritmom vatrometa u kombinaciji s K-sredinama 99
6.2 Rezultati prilagodjavanja algoritama za prostor cjelobrojnih rješenja . 111
6.2.1 Segmentacija slike mozga zasnovana na algoritmu krijesnica u kombinaciji s algoritmom grupiranja K-sredina 111
6.3 Rezultati prilagodjavanja algoritama za različite strukture rješenja . . 119
6.3.1 Problem planiranja putanje bespilotnih letjelica prilagođenim algoritmom krda slonova 119
6.3.2 Prilagođeni algoritam umjetne kolonije pčela za trokut minimalne težine 127

7 Zaključak 134

8 Literatura 138

1 Uvod

Veliki broj problema koji se javljaju u svakodnevnom životu, bilo u industiji, znanosti ili u životu pojedinca, mogu se definirati kao problemi optimizacije. Ti svakodnevni problemi često su teški problemi optimizacije. Rješavanje zadataka optimizacijskih problema čini raznovrsni znanstveni pristup u nastojanju da se pronađe koje rješenje je najbolje moguće, optimalno. U rješavanju tih problema važnu ulogu imaju mnogobrojne matematičke discipline kao što su matematička analiza, vjerojatnost, statistička teorija, teorija igara, linearno, nelinearno i dinamičko programiranje, heurističko-matematičko programiranje i dr. Većina teških problema optimizacije danas se rutinski rješava matematičkim metodama razvijenim tijekom stoljeća. Klasa problema, od kojih su mnogi od velike praktične važnosti, ostaje nerješiva. Poznati primjer takvog problema je problem putujućeg trgovca koji je ekvivalentan problemima optimizacije neprkidnih funkcija s ogromnim brojem lokalnih optimuma. Tijekom posljednjih nekoliko desetljeća metaheuristika nadahnuta prirodom uspješno se koristi za rješavanje teške probleme optimizacije. Algoritmi inteligencije rojeva istaknuta su klasa algoritama nadahnutih prirodom gdje rojevi jednostavnih pojedinaca koji interakcijom i razmjenom informacija pokazuju kolektivnu inteligenciju koja vodi ka optimalnim rješenjima.

Znanstveni pristup u donošenju odluka uključuje obično upotrebu jednog ili više matematičkih modela. Na temelju stvarnog problema napravi se matematički model koji služi za izvođenje potrebnih analiza. Modeliranje problema može biti jednostavno ali i vrlo izazovna i neintuitivna. Na temelju tih analiza pokušava se doći do traženih odgovora u vezi s postavljenim problemom. Da bi neko rješenje bilo optimalno tj. najbolje moguće treba imati mjeru kojom se određuje njegova kvaliteta. Uvođenjem mjere, ne samo da se određuje njegova kvaliteta već se i omogućava njegova usporedba s drugim mogućim rješenjima. Mjeru kvalitete u matematičkom modelu predstavlja funkcija nazvana fitnes funkcijom, funkcijom cilja, objektna funkcija ili kriterijska funkcija. Funkcija cilja svakom rješenju pridružuje vrijednost uz pomoć kojega se vidi efikasnost izvođenja zadatka i njegov konačan rezultat.

Uspješnost optimizacije ovisi o ekstremnim vrijednostima funkcije cilja. Varijable koje treba odrediti obično su uvjetovane međusobnim relacijama i ograničenjima

koje čine skup ograničenja. Svako rješenje koje zadovoljava ta postojeća ograničenja naziva se dopustivim rješenjem. Budući da jedan optimizacijski zadatak može imati više dopustivih rješenje onda se govori o dopustivom skupu rješenja. Uspješnost određivanja optimalnog rješenja nekog optimizacijskog zadatka zavisi, prije svega, o osobinama funkcije cilja, o tipu ograničenja i primjenjenoj metodi. Osnovni zadatak u teoriji optimizacije je razvoj metoda za rješavanje zadataka globalne optimizacije.

Kako ne postoji jedinstveno rješenje odnosno jedan algoritam koji može rješavati sve probleme, već je potrebno algoritme prilagođavati razmatranom problemu, ova disertacija ima za cilj prilagođavanje i analiziranje algoritama inteligencije rojeva za primjene na praktične probleme sa različitim prostorima pretrage. Glavni akcenat je prilagodba algoritama za različite prostore pretraga, od prostora vektora realnih brojeva preko vektora cijelobrojnih vrednosti do kompleksnih struktura poput putanja i grafova. Analizom svojstava funkcije cilja kao i prostora rješenja, algoritmi inteligencije rojeva mogu se u većoj ili manjoj mjeri mijenjati kako bi se prilagodili za rješavanje konkretnog problema. Za pojedine probleme prilagodba ili adaptacija algoritma može uključivati samo jednostavno zaokruživanje rješenja na najbliži cijeli broj (razni lokacijski problemi i problemi iz područja procesuiranja digitalnih slika) dok u nekim situacijama da bi se algoritam prilagodio rješavanju problema, potrebno je u značajnoj mjeri izmijeniti formule generiranih rješenja, računanja bliskih rješenja, računanja funkcija cilja, itd. Pregledom literature može se zaključiti da je ova tema veoma aktivno područje istraživanja.

Cilj ove disertacije je ispitivanje mogućnosti i kvaliteta različitih prilagodbi algoritama inteligencije rojeva za različite prostore pretrage. Preciznije rečeno cilj ovog istraživanja je prilagođavanje i implementacija algoritama inteligencije rojeva za različite prostore pretrage i njihova upotreba za rješavanje praktičnih optimizacijskih problema. U okviru ove disertacije testirana su i uspoređena kvaliteta predloženih prilagodbi s metodama iz literature. Rad sadrži uvod, zaključak, spisak korišćene lirerature i pet poglavlja.

Posle uvodnog poglavlja, u drugom poglavlju definiran je pojam optimizacije kao znanstvene discipline. Dana je klasifikacija optimizacijskih problema i svaka klasa je pojedinačno opisana i definirana. Posebna pozornost je posvećena klasi NP-teških problema optimizacije.

Treće poglavlje bavi se algoritmima za rješavanje problema optimizacije. Izložen je i analiziran pregled njihove klasifikacije i uvedeni su pojmovi heuristike i metaheuristike.

Četvrto poglavlje posvećeno je posebnoj klasi prirodom inspiriranih algoritama koji se nazivaju algoritmi inteligencije rojeva. Na početku su navedene njihove zajedničke osobine ovih algoritama odnosno zajednička struktura, a zatim su pojedinačno opisani najznačajniji predstavnici algoritama koji pripadaju ovoj klasi i to: algoritam kolonijom mrava, algoritam za optimizaciju rojevima čestica, algoritam umjetne kolonije pčela, algoritam krijesnica, algoritam šišmiša, brain storm optimizacija, algoritam krda slonova i algoritam vatrometa s posebnim naglaskom na njegovu najjednostavniju verziju, algoritam ogoljenog vatrometa.

U petom poglavlju predstavljene su prilagodbe algoritama inteligencije rojeva za različite prostore pretrage i njihova primjena na praktične probleme. Primjena ovih algoritama na praktične probleme podrazumijeva da ih prije svega treba prilagoditi, modificirati, kombinirati, hibridizirati i podesiti za različite prostore pretrage. Primjene koje su spomenute u ovom poglavlju odnose se na prilagođavanje algoritama za optimiziranje stroja potpornih vektora i problem grupiranja podataka što su problemi gdje su rješenja vektori realnih vrijednosti, za pronalaženje optimalnih pragova za segmentiranje digitalnih slika što je problem gdje je rješenje vektor cjelobrojnih vrijednosti, za pronalaženje optimalne putanje robota gde je rješenje struktura koja predstavlja sekvencu točaka i za problem triangulacije minimalne težine gdje je rješenje spisak rubova koji su deo triangulacije.

U šestom poglavlju izloženi su rezultati postignuti prilagodbama algoritama inteligencije rojeva na praktične probleme. Prikazane su primjene algoritama inteligencije rojeva na pet različitih praktičnih problema. Razmatrani su sljedeći problemi: otkrivanje eritematoloških-skvamoznih bolesti pomoću optimiziranog stroja potpornih vektora, grupiranje podataka web inteligencije, segmentacija MRI slika mozga, problem planiranja putanje bespilotnih letjelica i problem pronalaska trijangulacije minimalne težine u dvodimenzionalnom prostoru. Prilagodbe svih algoritama inteligencije rojeva detaljno su opisane i obrazložene. Uz izložene eksperimentalne rezultate koji su dobiveni ili testiranjem na standardnim skupovima podataka ili testiranjem na vlastitoj zbirci instanci zbog nedostatka istih u literaturi (u slučaju

problema pronalaženja trijangulacije minimalne težine) dana je usporedba s drugim vodećim algoritmima u literaturi kao i detaljna analiza dobivenih rezultata.

Deo rezultata istraživanja prikazanih u ovoj disertaciji prezentovani su na međunarodnim konferencijama i objavljeni u zbornicima koji su indeksirani u znanstvenim bazama kao što su WoS, Scopus, IEEE, dok su određeni delovi rezultata objavljeni u međunarodnim časopisima indeksiranim na SCI listi.

2 Optimizacija

Optimizacija je jedna od najšire korištenih tehnika, ne samo u znanosti, nego i općenito u ljudskom životu. Praktički može se reći da svaka stvar koju ljudi rade predstavlja neku optimizaciju - počevši od praljudi i njihovih lovačkih vještina pri hvatanju životinja pa sve do danas do problema kao što je dizajniranje svemirskih raketa. Ono što je zanimljivo jest da je kroz povijest optimizacija uvijek spadala u matematičko područje. Problemi optimizacije koji su od značajnog interesa za stvarni život mogu biti jednostavni ili pak veoma kompliciranu za rješavanje pa samim tim uključuju i različite razine matematike. Za opću populaciju obično metode za rješavanje problema optimizacije predstavljaju kompliciranе i teške matematičke tehnike.

Sa druge strane, ljudi cijeli svoj život kroz razne aktivnosti koriste optimizaciju pri čemu većina njih i nije svjesna da to što rade jest upravo optimizacija. Puno je primjera iz života gdje je svakodnevno koristimo: prilikom kupovine pri odabiru najprikladnijeg proizvoda, pronalaženje najbržeg ili najkraćeg puta do nekog odredišta, ostvarivanje maksimalnih prinosa prilikom investicija, i mnogi drugi primjeri. Zaključak je da u većini aktivnosti koje čovjek obavlja, on se trudi optimizirati ograničene izvore (novac, vrijeme, i sl.) kako bi ih iskoristio na najbolji mogući način. Pod optimizacijom se podrazumijeva pronalaženje najboljeg rješenja u zadanom kontekstu u kojemu se nalazi mogući skup rješenja, definirani ciljevi i ograničenja. Dakle kao aktivnost, optimizacija je svuda oko nas, koristi se u svim granama gospodarstva od ekonomije, transporta, sustavima za navodnjavanje, procesu distribucije električne energije, itd.

Cilj optimizacije je pronaći rješenje nekog problema koje je ili optimalno ili blizu optimalnog u odnosu na moguća ograničenja i postavljene ciljeve jer promatrani životni problemi mogu posjedovati više parametara čime utječu na samu suštinu rješenja. Prema tome, optimizacijom nekog problema dolazi se do kombinacije parametara koji najbolje odgovaraju. U konačnici se često dobiva veći broj mogućih alternativa rješenja. Iz cijele skupine mogućih rješenja nastoji se pronaći najbolje, ili u ovisnosti o prihvaćenim uvjetima, optimalnu alternativu. Naime, optimizacija je složen problem povezan s višestrukim ciljevima i velikim brojem promjenjivih veličina.

Proces optimizacije sastoji se od nekoliko koraka koji se izvršavaju jedan za drugim: identificiranje i definiranje problema, konstruiranje i rješavanje modela problema, implementacija i evaluacija rješenja. Izbor mogućih alternativa odnosno rješenja problema određeno je željom pojedinca da donese najbolju moguću odluku. Mjera kvalitete odluke definira se ciljem koji se želi postići, u matematičkom jeziku, funkcijom cilja.

U starijim vremenima računanje površine područja kruga predstavljao je težak problem dok danas, s razvojem i napretkom matematike, industrije i tehnologije, kompliciraniji zadatci se javljaju kada se na primjer trebaju napraviti zrakoplovi i svemirske rakete, tada matematika postaje kompliciranija. Optimizacija se smatra vrlo teškim matematičkim problemom. Međutim, matematika se brzo razvijala tijekom posljednjih nekoliko stoljeća, a većina tih teških problema optimizacije se danas rutinski rješavaju brojnim matematičkim metodama.

Danas se razlikuje klasa problema optimizacije gdje su problemi na prvi pogled prilično naivni, za koje je vrlo jasan algoritam za njihovo rješavanje koji je najčešće toliko jednostavan da ga mogu shvatiti i učenici u osnovnoj školi, međutim problem je što izvršavanje tog algoritma zahtijeva previše vremena (na primjer stotine i tisuće godina). Ovoj klasi problema pripada i poznati problem putujućeg trgovca - trgovac treba obići određeni broj gradova, posjećujući svaki grad točno jednom, uz minimalne troškove. Algoritam za rješavanje ovog problema je jasan, ispitati sve moguće putanje (permutacije broja gradova) i izabrati najbolju. Problem je što za već mali broj gradova npr. 20 gradova, izvršenje algoritma zahtijeva previše vremena, gotovo 4000 godina, što predstavlja potpuno neupotrebljiv rezultat. Postoje brojni problemi ove vrste, kominatorni problemi i oni pripadaju grupi NP-teških problema.

Ekvivalentno ovakvim kombinatornim problemima postoje i NP-teški problemi u kontinualnom prostoru. Tražeći globalni minimum neprekidnoj funkciji, uvidjelo se da mnoge funkcije imaju enormni broj lokalnih minimuma. Iako se njihovom rješavanju drugačije pristupa s vrlo različitim matematičkim tehnikama, uvidjelo se da je problem pronalaska koji od lokalnih minimuma je bolji od svih drugih zapravo je ekvivalentan problemu putujućeg trgovca. Potrebno je ispitati sve lokalne optimume i pronaći najbolji među njima, a jednostavno toliko puno vremena nema.

Ovo predstavlja unikatnu situaciju obzirom da sa jedne strane nakon toliko godina

razvoja matematike teški problemi optimizacije se rutinski rješavaju, dok se s druge strane s problemima koji su tako jednostavni, zvuče vrlo jednostavno i algoritam rješavanja je jednostavan ali praktično neupotrebljiv, čovjek ne zna nositi. A upravo se s takvom vrstom problema čovjek susreće svakodnevno te je primoran iznaći način kako se s njima nositi.

U jednom trenutku se nanovo kreće od samog početka razvoja područja optimizacije, za pronalaženje metoda za rješavanje tih problema koji su od velike praktične vrijednosti ali nerješivi poznatim metodama. Za pronalaženje rješenja ovakvih problema, kreće se s naivnom metodom, metodom slučajnog pretraživanja. Generira se nasumično određen broj rješenja i od tih rešenja se odabere najbolje. Ideja je da se uz pomoć slučajne pretrage nasumično ispita tisuću različitih pozicija. Rezultat je da ponekad će se dogoditi da će dobre točke biti pogođene, ali to i ne mora biti slučaj. Ovo jest jedan od načina na koji se može nositi s ovakvim problemima, međutim veoma je nepouzdan i nepredvidiv. Čak i za najjednostavnije problema verovatnoća pronalaženja optimalnog rješenja je minimalna iako raste sa brojem generiranih rješenja.

Idući pristup u rješavanju ovog problema ne koristi jedino metodu slučajnog pretraživanja već vođenu metodu slučajnog pretraživanja. Ovo znači da rješavanje problema je potrebno početi s metodom slučajnog pretraživanja, a zatim neku od informacija upotrijebiti za daljnje vođenje pretrage. Uz pomoć samo dva mjerenja moguće je odrediti smjer niz brdo i pretragu dalje nastaviti u tom smeru. Na ovaju način metoda slučajnog pretraživanja postaje vođena metoda slučajnog pretraživanja. U mnogim slučajevima, budući je okolina objektne funkcije nepoznata, ova metoda upada u prvi lokalni minimum te ostaje tamo zaglavljena. Da bi se uspjelo izaći iz tog problema, prave se mjerenja u klasterima. Najbolje rješenje iz svakog klastera uzima se za njegovog predstavnika. Na taj način dolazi se do globalnog silaska nizbrdo, ne više lokalnog.

Naravno, uvijek je moguće naći kompliciranije i još više kompliciranije funkcije za koje je potrebno pronalaziti sve bolje i bolje tehnike rješavanja takvih prepreka. Takvo razmišljanje je u redu, ali konačni rezultat tog razmišljanja je bespomoćnost. Problemi postaju sve kompliciraniji, metoda slučajnog pretraživanja je odviše kruta, vođena metoda slučajnog pretraživanja je bolja ali ne zna se koji je pravi način

vođenja odnosno ne postoji univerzalan princip vođenja ka boljem rješenju.

I po prvi puta u povijesti matematike ili povijesti računalnih znanosti ne vidi se drugi način u rješavanju tih problema već se matematičari prepuštaju prirodi. Nema više strogih i precizno definiranih metoda već se okreću prirodi i pokušavaju oponašati procese i pojave u njoj, jer priroda je efikasna. Stoga se način kako voditi proces metode slučajnog pretraživanja pokušava pronaći i simulirati iz prirode. Bila je to doista revolucionarna ideja s kojom su se dobila dobra rješenja za teške probleme optimizacije, a što je još najvažnije, do tih rezultata se dolazi u veoma kratkom vremenu.

2.1 Klasifikacija problema optimizacije

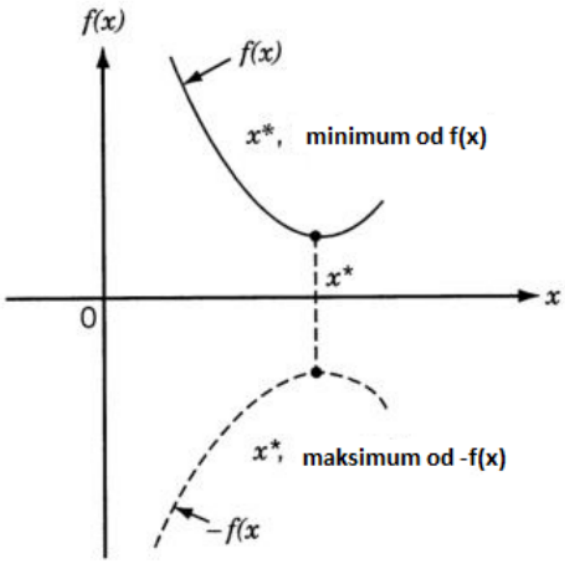
Matematički, problemi optimizacije se mogu definirati na sljedeći način. Bez smanjenja općenitosti, za optimizaciju se promatra minimizacija i matematički možemo prikazati kao:

$$\text{za datu funkciju } f : R^n \to R, \text{ treba naći } x^* \in R^n,$$
$$\text{tako da vrijedi } \forall x \in R^n,\ f(x^*) \leq f(x)$$

gdje je R^n prostor pretraživanja funkcije f, n varijabla koja označava dimenziju prostora pretraživanja, a x^* oznaka za optimalno rješenje. Ako točka x^* odgovara minimumu vrijednosti funkcije $f(x)$, ista točka također odgovara maksimalnoj vrijednosti od negativne te iste funkcije, $-f(x)$. To je razlog, da se bez smanjenja općenitosti, može optimizaciju smatrati minimizacijom budući da se minimum od te funkcije može naći kao maksimun od negativne iste te funkcije, slika 1.

Naziv pod kojim se još ovaj prostor može naći u literaturi je prostor varijabli budući da se traži skup odgovarajućih vrijednosti varijabli. U slučaju problema optimizacije funkcije, n označava broj varijabli.

S obzirom da zadatci koji se postavljaju kao problemi optimizacije dolaze iz brojnih različitih oblasti znanosti, ti zadatci su vrlo raznovrsni pa su i problemi optimizacije koji ih opisuju različiti. Kako bi se pristupilo rješavanju ovih problema i pravljenju heuristika i/ili metaheuristika za njihovo rješavanje, potrebno je imati



Slika 1: *Minimum od f(x) je jednak maksimumu od -f(x)*

određeno znanje o problemima koji se rješavaju. Iz ovoga se može zaključiti da je klasifikacija problema optimizacije veoma bitna. Klasifikacija problema optimizacije je relativno kompliciran zadatak s obzirom da se klasifikacija može vršiti prema brojnim elementima. Samim tim se u literaturi može pronaći veoma veliki broj klasifikacija problema optimizacije. Ovo je opet, još jedan od pokazatelja širine polja optimizacije.

U [10] je izgrađen podskup problematičnih klasa koje se mogu osnovati suprostavljanjem određenih aspekata općeg problema optimizacije. Neka je

$$
\begin{aligned}
x \in R^n, \text{ vektor od varijabli odluke } x_j, j = 1, 2, \ldots, n;\\
f : R^n \to R \cup \{\pm\infty\} : \text{ ciljna funkcija;}\\
X \subseteq R^n : \text{ osnovni skup definiran logički/fizički;}\\
g_i : R^n \to R : \text{ funkcije ograničenja koje definiraju restrikcije na x :}\\
g_i(x) \geq b_i, \ i \in I; \text{ nejednakosti ograničenja}\\
g_i(x) = d_i, \ i \in E \text{ jednakosti ograničenja.}
\end{aligned}
$$

gdje $b_i \in R, (i \in I)$ i $d_i \in R, (i \in E)$ označavaju desne strane ovih ograničenja;

bez gubitka općenitosti zapravo se može dopustiti da su sve jednake nuli jer se sve konstante mogu ugraditi u definicije funkcija $g_i, (i \in I \cup E)$. Iz definicije optimizacije se vidi da vrsta problema ovisi o prirodi funkcija f i g_i i skupa X i ovisno o njima slijedi

- Linearno programiranje (eng. linear programming, LP) funkcija cilja f je linearna, a funkcije ograničenja g_i su afine funkcije.
- Nelinearno programiranje (eng. nonlinear programming, NLP) neke od funkcija f i g_i su nelinearne.
- Neprekidna optimizacija (eng. continuous optimization) f i g_i su neprekidne funkcije na otvorenom skupu koji sadrži X, X je zatvoren i konveksan.
- Cjelobrojno programiranje (eng. integer programming, IP) $X \subseteq 0, 1^n$ ili $\subseteq Z^n$.
- Optimizacija bez ograničenja (eng. unconstrained optimization) $I \cup E = \emptyset$; $X = R^n$.
- Optimizacija s ograničenjima (eng. constrained optimization) vrijedi $I \cup E \neq \emptyset$ i/ili $X \subset R^n$.
- Diferencijabilna optimizacija (eng. differentiable optimization) f i g_i su bar jednom neprekidne diferencijabilne funkcije na otvorenom skupu koji sadrži X, X je zatvoren i konveksan.
- Nediferencijabilna optimizacija (eng.non-differentiable optimization) bar jedna od f i g_i je nediferencijabilna.
- Konveksno programiranje (eng. convex programming) f je konveksna funkcija; $g_i, (i \in I)$ su konkavne; $g_i (i \in E)$ su afine; X je zatvoren i konveksan.
- Nekonveksno programiranje (eng. non-convex programming) je komplement od konveksnog programiranja.

Od gore navedenih, dvije najvažnije klase problema su linearno i nelinearno programiranje, od kojih se

- linearno programiranje smatra "lagano"u smislu da postoje algoritmi koji mogu učinkovito riješiti svaki problem LP-a u praksi;
- nelinearno programiranje smatra se "teško"; u smislu da ne postoje algoritmi koji mogu učinkovito riješiti svaki problema NLP-a u praksi, područje problema NLP je veoma široko kao što obuhvaća veoma teške probleme obuhvaća i veoma lagane.

Ne postoji jedinstvena klasifikacija problema optimizacije. Problemi optimizacije značajno se razlikuju po svojim svojstvima. Od mnogih navedenih podjela u literaturi spominje se i sljedeća koja se temelji na [45]:

- ograničenjima - problemi optimizacije sa ograničenjima i problemi optimizacije bez ograničenja
- prirodi varijabli - statički problemi optimizacije i dinamički problemi optimizacije
- fizičkoj strukturi problema - optimalna kontrola problema i neoptimalna kontrola problema
- prirodi uključenih jednakosti - nelinearni problemi programiranja, linearni problemi programiranja, geometrijski problemi programiranja i kvadratni problemi programiranja
- dopustivim vrijednostima varijabli - problemi programiranja s cjelobrojnim vrijednostima i problemi programiranja s realnim vrijednostima
- odvojivosti funkcija - odvojivi problemi programiranja i neodvojivi problemi programiranja
- broju ciljnih funkcija - jednokriterijski problemi i višekriterijski problemi

U ovom radu, uzimajući u obzir već definirane općenite karakteristike problema optimizacije, klasifikacija se vrši na osnovi [118] gdje su zadani sljedeći kriteriji:

1. Broj i vrsta funkcija cilja

   Problemi optimizacije se po ovom kriteriju dijele na probleme koji imaju

definiranu samo jednu funkciju cilja (tj. imaju jedan kriterij optimalnosti) i na probleme koji imaju više funkcija ciljeva. Problemi s jednim kriterijem optimalnosti nazivaju se još i jednokriterijski problemi (engl. single objective problems), dok su oni drugi višekriterijski problemi (engl. multi-objective problems).

Vrlo je važno znati o kojem je problemu optimizacije po ovom kriteriju riječ. Budući da su različite implementacije optimizacijskog algoritma koji za konačni cilj ima jedan broj (jednokriterijski problem) odnosno Pareto plohu (višekriterijski problem). U slučaju kada se radi o višekriterijskom problemu potrebno je za svaki kriterij točno odrediti je li on maksimizacija ili minimizacija pripadne funkcije cilja.

Postoje i problemi optimizacije koji nemaju funkciju cilja. Oni čine veoma specifičnu vrstu optimizacijskih problema koji zadovoljavaju ograničenja (eng. constraint satisfaction problems). Postoji mogućnost prilagođavanja ove vrste problema na jednokriterijske probleme na način da im vrijednodnost funkcije cilja bude broj prekršenih ograničenja. Naravno, potrebno je minimizirati tu vrijednost na nula.

2. Broj i vrsta varijabli

   Skup varijabli predstavlja ulaznu veličinu svakog problema optimizacije koji za cilj ima odrediti njihove optimalne vrijednosti. Vrsta varijable se u tom skupu može klasificirati: cjelobrojna i realna varijabla, permutacija, niz varijabli, graf, matrica, itd. U optimizacijskom algoritmu svaka varijabla je reprezentirana nekom programskom strukturom koja čini element programskog jezika. Važnost vrsta varijabli nad kojima je definiran problem vidi se u postojanju zasebnih grana teorije optimizacije vezanih uz diskretne, kombinatorne i kontinuirane probleme.

3. Prisutnost ograničenja

   Optimizacijski problemi se po ovom kriteriju dijele na one koji imaju definirana ograničenja i one koje takvih ograničenja nemaju. U slučaju da su takva ograničenja definirana, konačno rješenje ih mora zadovoljiti za razliku od slučaja kod kojih ograničenja nema i gdje su moguća sva rješenja iz skupa

rješenja. Naravno da je implementacija takvog algoritma uvelike jednostavnija za razliku od problema s definiranim ograničenjima.

4. Prisutnost parametara problema

   Vrijednosti parametara problema za pojedinačno provođenje optimizacije su konstantne. Njihova prisutnost u optimizacijskim problemima veoma je važna jer omogućava jednostavno provođenje istraživanja kako o njihovoj promjeni ovisi optimalno rješenje (analiza osjetljivosti). Također uz pomoć njih mogu se raditi eksperimentiranja za primjenu algoritama inteligencije rojeva na praktične probleme što će se kasnije spomenuti.

2.2 Optimizacija s jednom ili više funkcija cilja

Bitno je da svaki optimizacijski problem ima definiran jedan ili više kriterija optimalnosti po kojem ili po kojima će se ocijeniti kvaliteta pojedinih rješenja. U literaturi i praksi obično se takvi kriteriji nazivaju funkcijama cilja (engl. objective function). Kod jednokriterijskih problema funkcija cilja definira se kao preslikavanje:

$$f : S \rightarrow R, \tag{1}$$

gdje S predstavlja Kartezijev produkt skupova nad kojima su definirane pojedinačne varijable problema.

Funkcija cilja za izlaz ima jedan (najčešće realni) broj koji predstavlja mjeru kvalitete rješenja. Ako se radi o problemu minimizacije (maksimizacije), cilj optimizacije je naći one vrijednosti ulaznih varijabli za koje će vrijednost funkcije cilja biti minimalna (maksimalna).

U civilnom inženjerstvu za funkciju cilja obično se uzima minimizacija troškova. U mehaničkom inženjerstvu, maksimizacija mehaničke uspješnosti je logičan izbor za ciljnu funkciju. U zrakoplovnom strukturnom dizajnu za ciljnu funkciju minimizacije generalno se uzima težina.

Puno je kompliciranije definirati cilj optimizacije u slučaju optimizacijskog problema s više definiranih kriterija optimalnosti, odnosno kada imamo dvije ili više definiranih funkcija cilja (eng. multi-objective optimization, MOO). Naime, dolazimo do situacija u kojoj je jedno rješenje bolje po jednom kriteriju, a neko drugo rješenje

je bolje s obzirom na neki drugi kriterij. Vrlo malo je vjerojatno da će se baš u točno jednom te istom rješenju istovremeno postići optimalne vrijednosti svih funkcija cilja.

Pristup težinske sume (eng. weight sum) je u literaturi jedan od najšire korištenih načina za rješavanje MOO problema. Ova metoda spaja sve ciljeve u jedan, na način, da množi vrijednost svakoga cilja s odgovarajućim težinskim koeficijentom. Dilema do koje dolazimo jest koju vrijednost težinskog koeficijenta dodijeliti svakom cilju. U slučaju vrlo različitih numeričkih vrijednosti radi se skaliranje.

Za svaki cilj $f_1(x), ..., f_n(x)$ bira se skalarni težinski koeficijent ω_i, tako da se rješavanje problema svodi na optimizaciju složene funkcije cilja U [75]:

$$U = \sum_{i=1}^{n} \omega_i f_i(x) \tag{2}$$

U literaturi se spominje i metoda hijerarhije ciljeva kao još jedan način za rješavanje MOO problema. Iz samog naziva se može zaključiti da se određuju prioriteti za svaki cilj. Optimizira se prvo najvažniji cilj (s najvećim prioritetom) i tek nakon njegove optimizacije dolazi na red optimizacija drugog, pa trećeg, itd. cilja. Ovaj pristup ima veliki broj praktičnih primjena u literaturi [74].

Također i koncept Pareto optimalnosti je jedan od šire korištenih načina za rješavanje MOO problema, možda jedan i od najpreciznijih. MOO problem matematički se može formulirati kao [22]:

$$\min\ [f_1(x), f_2(x), ..., f_n(x)],\ x \in S, \tag{3}$$

gdje je f skalarna funkcija, $n > 1$, a S je skup ograničenja određen jednadžbom:

$$S = \{x \in R^m : h(x) = 0,\ g(x) \geq 0\} \tag{4}$$

Ukratko po definiciji Pareto optimalnosti, kaže se da je vektor $x^* \in S$ Pareto optimalan ako svi ostali vektori $x \in S$ imaju veću vrijednost za barem jednu funkciju cilja $f_i, i = 1, ..., n$, ili imaju jednaku vrijednost za sve ostale funkcije cilja kao i rješenje koje je postavljeno da je Pareto optimalno [29].

Znači, rješenje je Pareto optimalno ukoliko ne postoji rješenje koje bi bilo strogo bolje po jednom kriteriju optimalnosti a da je istovremeno barem jednako dobro po svim ostalim kriterijima optimalnosti. Osim u slučaju kad postoji jedno rješenje koje

je najbolje po svim kriterijima optimalnosti, rješenje višekriterijskog optimizacijskog problema predstavlja više Pareto optimalnih rješenja. Skup tih rješenja naziva se Pareto skupom i njegovo nalaženje predstavlja konačni cilj kod višekriterijske optimizacije.

Za točku x^* kaže se da je slabi Pareto optimum ili slabo efikasno rješenje MOO problema ako i samo ako je $\nexists x \in S$ takvo da je $f_i(x) < f_i(x^*)$, za $\forall i \in \{1, ..., n\}$. S druge strane, točka x^* je jak Pareto optimum ili jako efikasno rešenje MOO problema ako i samo ako vrijedi da $\nexists x \in S$ takvo da je $f_i(x) \leq f_i(x^*)$, za $\forall i \in \{1, ..., n\}$, s makar jednom striktnom nejednakosti. U literaturi se još mogu naći kao pojmovi dominiranog i nedominiranog rješenja višekriterijskog optimizacijskog problema.

Danas je MOP veoma interesantna tema za radove koja je ujedno i veoma tražena. Veliki broj svakodnevnih problema su upravo zadani kao ciljevi koji se međusobno sukobljavaju i pitanje je kako zadovoljiti takve ciljeve. Sa stajališta optimizacije Pareto front je konačno rješenje. Međutim, s praktičnog stajališta na kraju cijelog procesa je korisnik koji treba izabrati što želi, a veliki je problem, što u većini slučajeva ne zna kriterije odabira.

2.3 Optimizacija bez ograničenja

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

Problemi globalne neprekidne optimizacije bez ograničenja (eng. continious global unconstrained optimization) formuliraju se na sljedeći način:

$$\min \ f(x), \ x = (x_1, x_2, ..., x_n) \in S \subseteq R^n, \tag{5}$$

gdje je $S \subseteq R^n$ prostor pretraživanja, a n dimenzija problema. S je n-dimenzioni hiper-kvadar u prostoru R^n koji se definira pomoću gornjih i donjih granica vrijednosti parametara:

$$lb_i \leq x_i \leq ub_i, \quad 1 \leq i \leq n, \tag{6}$$

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

2.4 Optimizacija s ograničenjima

Problemi s kojima se ljudi susreću kroz svoj život daleko su od jednostavnih. Malo koji nema neka definirana ograničenja. Njihova prisutnost uvelike komplicira izradu optimizacijskog postupka.

Prostor mogućih rješenja više nije predstavljen kompletnim Kartezijevim produktom skupova nad kojima su definirane varijable već se on značajno sužava. Rješenja koja zadovoljavaju takva ograničenja nazivaju se dopustiva (eng. feasible), dok rješenja koja ne zadovoljavaju ta ograničenja nazivaju se nedopustiva (eng. infeasible).

Kod manjeg broja problema s ograničenjima postoji mogućnost njihovog uklanjanja prikladnom reprezentacijom. Naprimjer, kod problema trgovačkog putnika, jedino ograničenje koje taj problem ima je da se svaki grad može posjetiti točno jedanput. Ukoliko se za reprezentaciju rješenja (odnosno varijabla nad kojom je definiran problem) uzme skup matrica n x n s elementima iz skupa $\{0, 1\}$ gdje vrijednost elementa $x_{i,j} = 1$ povlači da se iz i-tog grada ide u j-ti grad, očigledno je da se dodatno moraju definirati ograničenja da bi se ispunio uvjet da se iz svakog grada po jednom odlazi i u svaki grad se jednom dolazi (suma po svakom retku i po svakom stupcu mora biti jednaka 1). Međutim, ukoliko se za reprezentaciju rješenja uzme skup permutacija Sn (skup svih permutacija niza brojeva od 1 do n, za datu permutaciju imamo redoslijed kojim se posjećuju gradovi), automatski su ta ograničenja zadovoljena i na sva moguća rješenja mogu se primijeniti optimizacijski postupci optimizacije bez ograničenja.

Također, i kod nekih drugih problema kombinatorne optimizacije primjenom prikladne reprezentacije mogu se ukloniti eksplicitna ograničenja. U problemima gdje to nije moguće provesti, ograničenja se moraju staviti u samu definiciju optimizacijskog problema.

Problemi kombinatorne optimizacije s ograničenjima (eng. combinatorial constrained optimization) definiraju se skupom od tri elementa (S, f, Ω), gdje imamo S kao prostor pretrage, f kao funkciju cilja koju treba minimizirati ili maksimizirati, i Ω kao skup ograničenja koja dobivena rješenja moraju zadovoljiti da bi bila dopustiva [84]. Cilj je pronaći globalno optimalno rješenje s^* koje ima najmanju ili najveću vri-

jednost funkcije cilja (u ovisnosti treba li funkciju cilja minimizirati ili maksimizirati), a da su pritom ispunjena sva ograničenja data u formulaciji problema.

U literaturi se susrećemo s velikim brojem metoda rješavanja ograničenja (eng. constraint - handling methods). Među raznim metodama rješavanja ograničenja susrećemo se s metodama koje funkciju cilja i ograničenja kombiniraju u jednu funkciju (eng. penalty function), do metoda koje rade potpuno suprotno, odvajaju funkciju cilja od ograničenja. Jedna od najčešće korištenih metoda rješavanja ograničenja, koja se temelji na pravilima izvodljivosti je metoda koju je predložio Deb. Metoda je pokazala obećavajuće performanse u suočavanju s ograničenjima i koristi tri pravila izvodljivosti:

- Svako izvedivo rješenje je poželjnije od bilo kojeg neizvedivog;
- Među dva izvediva rješenja, poželjnije je ono koje ima bolju fitnes vrijednost;
- Ako su oba rješenja neizvediva, preferira se ono s nižom kaznom.

Elementarnim matematičkim operacijama tri različite vrste ograničenja $h(x) \leq 0$, $h(x) \geq 0$ i $h(x) = 0$ mogu se svesti na jedan oblik $g(x) \leq 0$ koji se i navodi u literaturi prilikom definiranja optimizacijskog problema.

Ograničenja jednakosti $h(x) = 0$ čine optimizaciju znatno težom, budući da prostor pretrage postaje veoma mali u usporedbi s cijelim prostorom pretrage. Pronaći bilo koje dopustivo rješenje postaje veliki problem. Da bi se to uspjelo, ograničenja jednakosti se zamjenjuju s ograničenjima nejednakosti. Jedna od novijih tehnika, ε metoda za upravljanje ograničenjima, predložena je od Takahama i Sakai [100]:

$$|h(x)| - \varepsilon \leq 0, \tag{7}$$

gdje je ε tolerancija kršenja ograničenja. Suština metode sastoji se u tome da se oko hiperravnine uvodi margina širine ε, unutar koje se rješenja smatraju dopustivima. Postupnim smanjivanjem te margine, kroz iteracije, na kraju su dobivena tražena rješenja.

2.5 Teški optimizacijski problemi

Algoritam je alat ili procedura za rješavanje nekog jasno određenog problema izračunavanja. On prima skup vrijednosti na ulazu i u nizu postupnih koraka izračunavanja ih obrađuje kako bi na kraju dao izlazne vrijednosti tj. rezultat. Zadatak algoritma je da odredi i opiše računarsku proceduru koja postiže željeni odnos ulaznih i izlaznih vrijednosti. Algoritam mora biti točan, što znači da mora rješavati definirani problem [31]. Dva najvažnija izvora koja algoritam ima na raspolaganju su vrijeme (vremenska složenost) i prostor (prostorna složenost). Pod pojmom prostorna složenost algoritma misli se na veličinu memorije potrebne da se izvrše izračunavanja. Zanemaruje se prostor za skladištenje podataka čime je omogućen proces uspoređivanja različitih algoritama za rješavanje istog problema. Pod pojmom vremenske složenosti algoritma misli se na broj koraka neophodnih za izračunavanje problema algoritma veličine n.

Na temelju minimalne prostorne i vremenske složenosti algoritma definira se težina problema [12]. Postoji cijela jedna teorija (eng. computational complexity theory) koja se bavi kategorizacijom problema prema njihovoj težini. Dvije osnovne klase problema odlučivanja su problemi koji pripadaju klasi složenosti P (polinomijalni) i klasi složenosti NP (nedeterministički polinomijalni). Problemi odlučivanja koji pripadaju klasi složenosti P mogu se riješiti u polinomijalnom vremenu. Problemi odlučivanja koji pripadaju klasi problema NP imaju karakteristiku da dobivenom rješenju zadanog NP-problema mogu provjeriti točnost u polinomijalnom vremenu.

Tokom proteklih stoljeća matematika se kao znanost razvijala i uspjela pronaći metode za rješavanje mnogih teških matematičkih problema. Također je i tehnologija toliko napredovala da današnja računala obavljaju na milione aritmetičkih operacija u sekundi. Međutim, računala nemaju toliku veliku moć rješavanja svakog problema u relativno kratkom razumnom vremenskom periodu. Postoje problemi koje nije moguće riješiti uz pomoć računala u razumnom vremenskom periodu. Jedan od primjera je i već spomenuti problem putujućeg trgovca, gdje je definiranje i implementiranje algoritma jednostavno za uraditi. Relativno za već mali broj gradova nije moguće izvršiti algoritam koji provjerava sva moguća rješenja u razumnom vremenskom periodu. Ovakvim problemima je skup rješenja toliko velik da računala ne uspijevaju ni u tisuću godina pronaći najbolje rješenje determinističkom metodom koja bi

provjeravala sva moguća rješenja. Njihov skup rješenja raste eksponencijalno pa unatoč impresivnom rastu tehnologije izrade procesora, ne može pomoći izračunavanju rješenja ovakvih problema.

Mnogi slični kombinatorni problemi, kao i velik broj optimizacionih problema iz realnog života pripadaju ovoj klasi NP-teških problema. Neki primjeri takvih problema su planiranje putanje robota (podmornice, bespilotne letjelice itd.), određivanje optimalnih lokacija u raznim sistemima, itd.

Početna ideja u pronalaženju rješenja za teške optimizacijske probleme je generaliziranje slučajnih rješenja. Najpoznatija takva metoda je Monte Carlo metoda koja između svih tih generiranih slučajnih rješenja bira najbolja rješenja i uzima ih kao aproksimizaciju optimalnog rješenja. Ako za ilustraciju uzmemo metodu Monte Carlo na n-dimenzionalnoj sferi koja je jednostavna funkcija s poznatim globalnim minimumom, vidimo da se već potencijalna rješenja generiraju na potpuno slučajan način, a za pronalazak kvalitetnog rješenja mora se generirati veliki broj točaka.

Slijedeća ideja u pronalaženju rješenja za teške optimizacijske probleme dolazi nekako na prirodan način. Proces slučajne pretrage navodi se ka optimalnom rješenju i time se želi osigurati generalizacija boljih rješenja. Taj proces navođenja ima osnovu u prethodnim iskustvima s čime se očekuje ubrzavanje procesa i poboljšavanje kvalitete optimizacijskog algoritma.

Za ilustraciju uzima se isti primjer n-dimenzionalne sfere, gdje se algoritam može usmjeravati prema nižim vrijednostima. Uz pomoć samo dva slučajno generirana rješenja može se odrediti smjer "nizbrdo"gdje se nova rješenja mogu generalizirati samo na tu stranu. S tim se značajno smanjuje broj rješenja koje je potrebno generirati i u relativno malom broju iteracija se može pronaći globalni optimum sa željenom preciznošću.

Nažalost, opisana heuristika se ne može uzeti za opće rješenje za sve optimizacijske probleme jer bi se najčešće zaglavljivala u lokalnom minimumu (maksimumu). Primjer za ilustraciju gdje ova metoda zaglavljuje u lokalnim optimumima je Rastrigin funkcija. Potrebno je naći novu heuristiku gdje bi se taj problem riješio, npr. pored navođenja nizbrdo potrebno je pronaći način da se pretražuju i drugi dijelovi prostora pretrage. Međutim, dolazi se do zaključka da će uvijek postojati neki kompleksniji primjer gdje ni ta nova heuristika ne bi bila dovoljno dobra. Uvidjelo se da je neophodno

slučajnu pretragu voditi, ali da nema adekvatnog i univerzalnog načina kako je treba voditi.

Na kraju se došlo do veoma neobične ideje. Proces vođenja slučajne pretrage prepušten je oponašanju nekih uspješnih procesa iz prirode. Pokazalo se da je to bila veoma uspješna ideja koja je otvorila vrata posve novoj disciplini u znanosti matematike.

3 Algoritmi za rješavanje problema optimizacije

Matematičke metode garantiraju pronalazak optimizacijskih rješenja za određene vrste funkcija. Ne postoji jedinstvena metoda za učinkovito rješavanje svih problema optimizacije. To je razlog konstruiranja niza metoda za rješavanje različitih vrsta problema optimizacije. Optimalne metode traženja poznate su i kao matematičke tehnike programiranja koje čine granu operativnog istraživanja. Operativno istraživanje se grubo sastoji od sljedećih područja [16]:

- metode matematičkog programiranja
- tehnike stohastičkih procesa
- statističke metode.

Metode matematičkog programiranja su korisne u pronalaženju minimuma funkcije s nekoliko varijabli ali pod propisanim skupom ograničenja. Tehnike stohastičkih procesa se koriste u analiziranju problema koji su opisani skupom slučajnih varijjabli poznate distribucije. Statističke metode se koriste u analizi eksperimentalnih podataka i u konstrukcijama empirijskih modela.

Početak postojanja optimizacije se može pratiti od Newton-a, Lagrange-a i Cauchy-a. Razvoj diferencijalnih metoda za optimizaciju bio je moguć zbog doprinosa Newtona i Leibnitza. Temelji proračuna varijacija postavili su Bernoulli, Euler, Lagrange i Weierstrasse. Ograničenu optimizaciju prvi je proučavao Lagrange, a pojam silaska uveo je Cauchy.

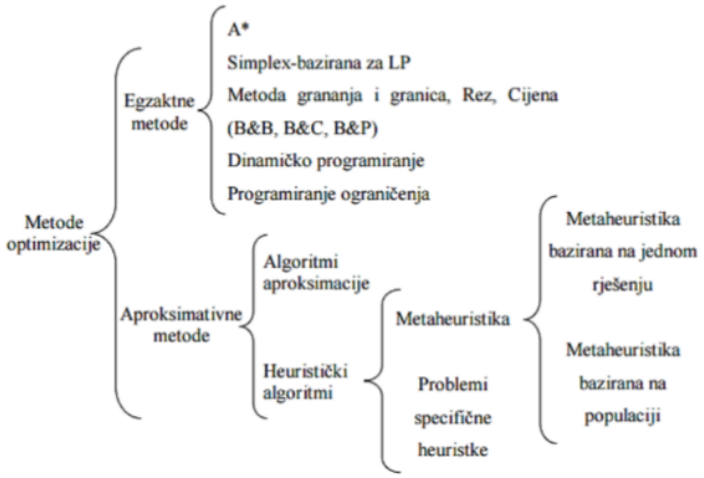
Unatoč tim prvim doprinosima, postignut je vrlo mali napredak do 20. stoljeća, kada je moć računala omogućila provedbu postupaka optimizacije, a to je naravno potaknulo daljnje metode istraživanja. U Velikoj Britaniji ostvareni su glavni pomaci na području numeričkih metoda neograničene optimizacije. Oni uključuju razvoj jednostavne metode (Dantzig, 1947), načela optimalnosti (Bellman, 1957), nužne i dovoljne uvjete optimalnosti (Kuhn i Tucker, 1951).

Većina problema planiranja, inžinjerskih i ekonomskih problema mogu se postaviti kao problemi optimizacije. Štoviše, također numerički problemi, kao što je problem rješavanja sustava jednadžbi ili nejednakosti, može se postaviti kao problem optimizacije.

U početku se istražuju problemi optimizacije u kojima su varijable odluke definirane u $\mathbb{R}^n$: neograničeni problemi optimizacije. Preciznije, proučava se problem određivanja lokalnih minimuma za diferencijabilne funkcije. Iako se ove metode rijetko koriste u aplikacijama, kao i u stvarnim problemima u kojima su varijable odluke podložne ograničenjima, tehnike u problemima neograničene optimizacije imaju značajnu ulogu u rješavanju općenitijih problema: poznavanje dobrih metoda za lokalnu neograničenu minimizaciju je neophodan preduvjet za rješavanje ograničenih i globalnih problema minimizacije.

3.1 Klasifikacija metoda optimizacije

Metode optimizacije se mogu klasificirati s različitih stajališta. Podjela metoda optimizacije je prema [101] na egzaktne metode i aproksimativne metode kao što je prikazano na slici 2.



Slika 2: *Metode optimizacije*

Egzaktne metode garantiraju optimalno rješenje ne vodeći brigu o računalnom vremenu. Aproksimativne metode se dijele na aproksimacijske algoritme i heurističke algoritme. Ove metode ne pronalaze nužno optimum. Aproksimacijski algoritmi pronalaze rješenje koje je zadane bliskosti optimumu dok s druge strane heuristički algoritmi ne davaju garanciju bliskosti rješenja kojeg pronalaze s optimumom međutim, za neke razrede problema heuristički algoritmi su jedini mogući za njihovo rješavanje.

Još jedna zanimljiva klasifikacija, koja se temelji na podacima funkcije koja će biti optimizirana, naime podjela je na:

- metode koje ne koriste derivaciju funkcije (izravno pretraživanje, ograničene razlike);
- metode temeljene na znanju prve derivacije (gradijent, konjugacijski pravci, kvazi-Newton);
- metode temeljene na znanju prve i druge derivacije (Newton).

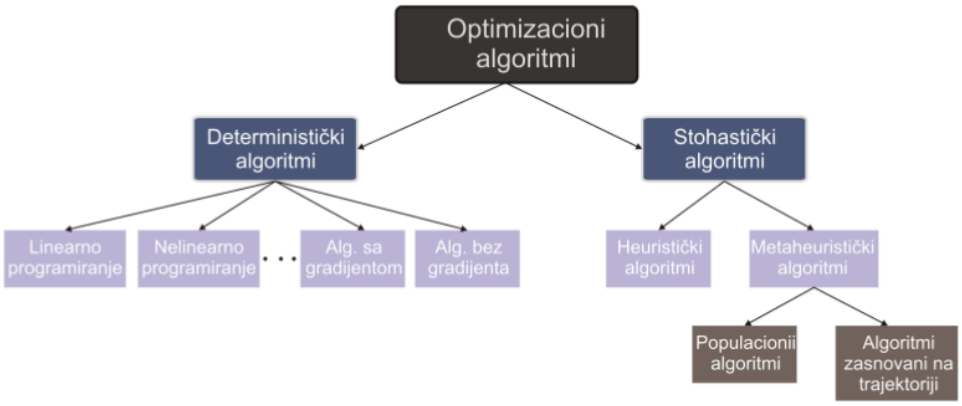
Spomenimo neke metode koje se najčešće koriste. Spuštanje gradijenta (eng. steepest descent method ili gradient descent) je algoritam prvog reda iterativnog optimiziranja za pronalaženje lokalnog minimuma funkcije. Da bi se pronašao lokalni minimum funkcije pomoću gradijentnog spuštanja, potrebno je praviti korake proporcionalne negaciji gradijenta (ili približnog gradijenta) funkcije u trenutnoj točki.

Znači, spuštanje gradijenta temelji se na opažanju da ako funkcija više varijabli $F(\mathbf{x})$ je definirana i diferencijabilna u susjedstvu neke točke $\mathbf{a}$, tada se funkcija $F(\mathbf{x})$ smanjuje najbrže ako kreće iz $\mathbf{a}$ u smjeru negativnog gradijenta od funkcije F u $\mathbf{a}$, $-\nabla F(\mathbf{a})$. Iz toga slijedi da, ako $\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$, za $\gamma \in \mathbb{R}_+$ dovoljno mali, tada $F(\mathbf{a_n}) \geq F(\mathbf{a_{n+1}})$. Drugim riječima, izraz $\gamma \nabla F(\mathbf{a})$ se oduzima od $\mathbf{a}$ zbog toga što se želimo kretati suprotno gradijentu, prema lokalnom minimumu. Imajući ovo na umu, započinje se s pretpostavkom $\mathbf{x}_0$ za lokalni minimum i smatra niz takav da $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ takav $\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0$. Dolaskom na monotonu sekvencu niza $F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots$ postoji mogućnost da niz $(\mathbf{x}_n)$ konvergira ka željenom lokalnom minimumu.

Newtonova se metoda primjenjuje na derivaciju f' od dvostruko diferencijabilne funkcije f kako bi se pronašla rješenja za $f'(x) = 0$, također poznata kao stacionarne točke od f. Ova rješenja mogu biti minimum, maksimum ili sedlasta točka. Središnji problem optimizacije je minimizacija funkcije. Razmatra se najprije slučaj funkcije s jednom realnom varijablom. S obzirom na dvostruko diferencijabilnu funkciju $f : \mathbb{R} \to \mathbb{R}$, nastojimo riješiti problem optimizacije $\min_{x \in \mathbb{R}} f(x)$. Newtonova metoda pokušava riješiti ovaj problem konstruirajući niz $\{x_k\}$ s početnom slučajnom točkom $x_0 \in \mathbb{R}$ koja konvergira prema minimumu x_* od f pomoću niza Taylorove aproksimacije drugog reda od f oko iterata. Taylorova ekspanzija drugog reda od f oko x_k je

$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2$. Sljedeći iterat x_{k+1} definiran je tako da se minimizira ta kvadratna aproksimacija u t i postavi se $x_{k+1} = x_k + t$. Ako je druga derivacija pozitivna, kvadratna aproksimacija je konveksna funkcija od t, i njegov minimum može se pronaći postavljanjem derivacije na nulu. Budući $0 = \frac{\mathrm{d}}{\mathrm{d}t}\left(f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2\right) = f'(x_k) + f''(x_k)t$, minimum se postiže za $t = -\frac{f'(x_k)}{f''(x_k)}$. Newtonova metoda izvodi iteraciju $x_{k+1} = x_k + t = x_k - \frac{f'(x_k)}{f''(x_k)}$. Naravno, da se ovo generizira u d-dimenziji zamjenom derivacije sa gradijentom a recipročna druga derivacija s inverzom Hessove matrice.

Optimizacijski algoritmi mogu se podijeliti na determinističke i stohastičke algoritme, kao na slici 3.



Slika 3: *Optimizacijski algoritmi*

Deterministički algoritam je algoritam koji će, s obzirom na određeni ulaz, uvijek proizvoditi isti izlaz, s tim da računalo uvijek prolaziti kroz isti slijed stanja. Poznati primjeri ovih algoritama su algoritam pretraživanja usponom, Newton-Raphsonova metoda bazirana na gradijentu, Simplex metoda, kao i niz drugih. S druge strane, stohastički algoritmi generiraju i upotrebljavaju slučajne varijable.

Heuristika je tehnika namijenjena bržem rješavanju problema kada su klasične metode prespore ili za pronalaženje približnog rješenja kada klasične metode ne mogu pronaći točno rješenje. Riječ heuristika je grčkog podrijetla i može se prevesti kao umjetnost otkrivanja novih strategija u rješavanju problema. To se postiže putem optimalnosti, potpunosti, točnosti ili preciznosti brzine [79]. Na neki način se mogu smatrati bržim putem.

Deterministički algoritmi su veoma lijepi u teoriji, ali većina praktičnih problema

se ne može rješavati uz pomoć njih u razumnom vremenu ili uopće. Heuristički algoritmi općenito uvijek ne garantiraju nalaženje optimalnog rješenja, ali pronalaze veoma dobro suboptimalno rješenje, koje se pronalazi veoma brzo, odnosno u većini slučajeva, u polinomijalnom vremenu. Stoga, za razliku od determinističkih, heuristički algoritmi su prihvatljiviji u rješavanjima praktičnih problema.

Metaheuristika je postupak više razine ili heuristički osmišljen da pronađe, generira ili odabere heuristički (djelomični algoritam pretraživanja) koji može pružiti dovoljno dobro rješenje problema optimizacije, posebno s nepotpunim ili nesavršenim informacijama ili ograničenim računalnim kapacitetom. Metaheuristika uzorkuje skup rješenja koji je prevelik da bi se mogao u potpunosti uzorkovati. Metaheuristika može dati nekoliko pretpostavki u vezi s problemom optimizacije koji se rješava, pa može biti korisna za niz problema. U riječi metaheuristika prefiks meta također potječe iz Grčke i znači metodologija višeg nivoa.

U usporedbi s algoritmima za optimizaciju i iterativnim metodama, metaheuristika ne jamči da se za neku klasu problema može naći globalno optimalno rješenje. Mnoge metaheuristike provode neki oblik stohastičke optimizacije, tako da pronađeno rješenje ovisi o skupu proizvedenih slučajnih varijabli. Pretraživanjem velikog skupa izvedivih rješenja, metaheuristika često može pronaći dobra rješenja s manje računskog napora nego algoritmi optimizacije, iterativne metode ili jednostavna heuristika. Kao takve, one su korisne taktike ili pristupi za probleme optimizacije.

Svakodnevni problemi s kojima se susrećemo u procesu optimizacije su različitog nivoa složenosti. Teoremi Nema besplatnog ručka (No- free-lunch theorems, NFL) naznačili su da za bilo koja dva algoritma A i B, ako A za neke probleme djeluje bolje od B, mora postojati neki problem na kojem će i B raditi bolje od A. Drugim riječima, ne postoji univerzalno bolji algoritam koji može biti učinkovit za sve probleme. NFL teoremi vrijede za deterministički ili stohastičku optimizaciju, gdje je skup parametara kontinualan, diskretan ili mješovit, a skup vrijednosti funkcije cilja je konačan.

Jedan od glavnih ciljeva rješavanja problema u praksi je pokušaj pronalaženja optimalnog ili kvalitetnog izvedivog rješenja u kratkom, prihvatljivom vremenskom rasponu. Za određenu vrstu problema, neki algoritmi doista mogu biti bolji od drugih. Iako teoretski solidan, NFL teorema u praksi može imati ograničen utjecaj jer ne

trebamo rješavati sve probleme, a također nam ne trebaju ni prosječne performanse. Jedan od glavnih ciljeva rješavanja problema u praksi je pokušaj pronalaženja optimalnog ili kvalitetnog dopustivog rješenja u kratkom, prihvatljivom vremenskom rasponu.

4 Algoritmi inteligencije rojeva

Inteligencija rojeva jedna je od grana metaheuristike nadahnute prirodom. U njima algoritmi simuliraju ponašanje insekta ili životinja koji u rojevima, kolonijama ili jatima rješavaju probleme s nekim ciljem npr. s ciljem pronalaskom hrane. Algoritmi inteligencije rojeva simuliraju nešto iz prirode gdje ima puno pojedinaca koji su svi vrlo jednostavni, ali svojom interakcijom u grupi mogu donositi pametne odluke tj. imaju sposobnost kolektivne inteligencije. U posljednjih 25 godina razvijeni su mnogi takvi algoritmi.

U simulaciji tih procesa iz prirode treba odlučiti koliko vjerno ih se želi slijediti. U osnovi, iz tog procesa u prirodi treba se izvući matematički model u kojem su bitna dva koraka ili dvije formule. Jedan korak je za eksploataciju ili intenzifikaciju, a drugi za eksploraciju ili diverzifikaciju. Procesom slučajnog pretraživanja dobivena su rješenja koja se žele i dalje koristiti. To predstavlja fazu eksploatacije. Koriste li se samo dobra rješenja, funkcija će zaglaviti u lokalnom minimumu. Potrebno je stvoriti dodatni mehanizam, kao protutežu, koji će omogućiti bijeg od lokalnih minimuma, što znači i odlazak do nekog ne tako obećavajućeg mjesta.

Uspjeh inteligencije rojeva je stvoriti dobru vezu između eksploatacije i eksploracije zbog koje će oni biti u ravnoteži. Ako je uključeno previše eksploatacije to će biti čisto matematička metoda. Ako ima previše eksploracije, bit će slična slučajnom pretraživanju te će trebati previše vremena za dobivanje bilo kakvog razumnog rješenja. Jednom kada se pronađe pravi matematički model on je zapravo uvijek samo djelić tog prirodnog fenomena. Na primjer, ako se promatraju mravi, koncentracija je samo na njihovom traženju kemikalije po zemlji na ničemu drugome.

Pitanje koje se postavlja je koliko daleko treba ići. Neki misle da je cijela ta priča o mravima previše naglašena i da bi se trebalo zapravo držati matematike, razvijati ta dva skupa formula s ciljem njihovog poboljšavanja, a ne razmišljati previše o prirodi fenomena. Druga strana razmišljanja je upravo suprotna. Više pažnje se posvećuje tom prirodnom dijelu, jer sve je započelo sa simuliranjem prirode i vjeruje se da priroda zna kako raditi određene stvari (jer ona to radi već milijunima godina). Zbog toga se nastoji da cijeli postupak simulacije bude što je moguće bliže prirodnom procesu.

Postoji još jedno zanimljivo gledište na ove procese. Umjesto gledanja na eksplo-

ataciju i eksploraciju, uvidjelo se da u svakoj generaciji postoje dva procesa. Jedan proces obuhvaća generiranje novih rješenja kandidata, dok se u drugom procesu donosi odluka koje od rješenja zadržati. Generiranje novih rješenja uključuje oboje, eksploataciju i eksploraciju ,a zatim je tu mehanizam koji će odlučiti koje od tih rješenja će se dalje koristiti.

Nakon što se napravi algoritam koji simulira određeni prirodni fenomen, slijedi teoretska faza u kojoj algoritam treba testirati, dokazati, itd. Dakle, prvo se ne radi na stvarnim aplikacijama, već se njihove izvedbe testiraju na referentnim (benchmark) funkcijama, što je dobro jer se algoritam može usporediti s drugim algoritmima. Međutim to može biti i veoma opasno jer referentne funkcije ne mogu pomoći u rješavanju praktičnih problema. Stoga, pristup koji je dobar za referentne funkcije nije baš dobar za stvarne probleme.

Zapravo, osnovni algoritam se smatra završenim ako je prošao dobro u fazi testiranja na benchmark funkcijama. Međutim, za praktične probleme treba napraviti dodatna prilagođavanja jer su oni obično ograničeni problemi. Postoji vrlo malo praktičnih neograničenih problema. Za ograničene probleme potrebno je dodati dodatne mehanizme za suočavanje s ograničenjima. Zanimljiva je činjenica da se kod ograničenih problema jako teško nositi s ograničenjima jednakosti. Zadovoljiti jednakost je veoma teško, praktički nemoguće. Dakle, s ograničenjima jednakosti nemoguće je dobiti bilo kakva dopustiva rješenja. Obično postoji mehanizam koji najprije tolerira opuštanje tih uvjeta. Dakle, umjesto da se zahtijeva da rješenja budu na hiperravnini, napravi se pojas oko hiperravnine čime se omogućava dobivanje nekih gotovo dopustivih rješenja. U procesu dalje, pojas se sužava, tako da konačno rješenje mora biti u hiperravnini, ali tamo se stiže postupno.

NP-teški optimizacijski problemi ne mogu se riješiti deterministčkim algoritmima u prihvatljivom vremenskom razdoblju. U današnje vrijeme postoje brojne tehnike i metode za njihovo rješavanje. Metaheuristika koja se temelji na nekim prirodnim procesima naširoko se koristi već desetljećima.

Tijekom posljednjih godina algoritmi nadahnuti prirodom, posebice algoritmi inteligentnih rojeva, uspješno su primijenjeni na tako teške optimizacijske probleme. Pored algoritama inteligencije rojeva, korišteni su i evolucijski algoritmi. Evolucijski algoritmi oponašaju evolucijski proces i primjenjuju operacije poput reprodukcije,

rekombinacije , mutacije i prirodne selekcije. Jedan od najprepoznatljivijih članova ove skupine algoritama je genetski algoritam [120]. Algoritmi inteligencije rojeva s druge strane oponašaju kolektivno ponašanje jednostavnih agenata iz prirode u njihovoj potrazi za hranom, izbjegavanju prepreka, neprijatelja, itd. Jedan od prvih algoritama inteligencije rojeva bio je algoritam optimizacije rojeva čestica [55] i optimizacije kolonijom mrava (ACO) [35], [115]. Tijekom posljednja dva desetljeća predloženi su brojni drugi algoritmi inteligentnih rojeva poput algoritma vatrometa [102], [110]; algoritma kolonije pčela [52]; algoritma šišmiša [136], [105], [6], [7]; pretraživanje kukavicom [138], [112]; algoritam krijesnica [134], [107]; optimizacija stada slonova [121], [106] i mnogi drugi. Njihova je upotreba povećana u rješavanju raznih NP-teških optimizacijskih problema u različitim oblastima gdje se pokazala vrlo uspješnom.

4.1 Osnove u algoritmima inteligencije rojeva

U posljednje dvije godine može se čuti da je ovo područje možda malo zasićeno i da postoji previše algoritama, previše istraživanja i promjena u razmišljanju o svemu tome. Na primjer, prije nekoliko godina gotovo su svi u radovima predstavljali fizičku osnovu algoritma. Danas je to gotovo zabranjeno. Dakle, ako u novinama govorite o mravima ili pčelama ili vatrometu, bolje je koristiti neutralne izraze kao što su rješenje, rješenje kandidata, bolje rješenje fitnes funkcija, a ne pčele, mravi, i tako dalje. Više to nije popularno, ali to je manje važan dio. Važan dio je da postoji tendencija ograničavanja cijelog ovog područja.

S praktičnog stajališta to nije opravdano. Iako se teoretski može ustvrditi da su svi ovi algoritmi slični i svi imaju eksploataciju i eksploraciju i mogu se transformirati jedan u drugi i tako dalje, obično u teoriji mogu biti gotovo jednaki, ali u praktičnom smislu, kada uzmemo u obzir troškove, mogu biti različiti. Na primjer, tako se može tvrditi da nam ne treba toliko različitih vrsta automobila, svi oni imaju kotače i motor i mogu se voziti bilo kada i bilo gdje. U nekom općem smislu to može i biti tako, međutim život u predgrađu, velikom gradu , malom gradu ili živjeti na farmi predstavlja različite izazove kojim se automobili moraju prilagoditi. Slična, ne ista, situacija je i sa ovim algoritmima. Nažalost za ljudsku rasu, a na sreću istraživača ne postoji niti jedan najbolji algoritam za sve probleme. Dakle, neki algoritmi će biti

bolji za neke probleme, a drugi za neke druge probleme. U tom je smislu, korisno imati različite algoritme.

Teoretski se može uzeti drugi algoritam i na kraju opet dobiti dobre rezultate, ali ako se želi obrada, bolje je imati više algoritama. U praktičnoj primjeni također su problemi iz prirode. U tom smislu, ima smisla napadati probleme iz prirode algoritmima iz prirode. Možda se ne zna točan mehanizam, ali oni su nekako prilagođeni jedni drugima. Teoretski pogled je da su svi algoritmi prilično ekvivalentni, ali to znači da su svi problemi prilično ekvivalentni. Ali to nije slučaj jer u prirodi postoje vrlo različito teški problemi. Međutim, iako su vrlo teški, obično su strukturirani na isti način. Imaju istu unutarnju strukturu. Možda je teško ili nemoguće znati koja je to struktura, ali ona postoji. A onda zapravo to nije više slučajni problem, koji možete napasti bilo kojom metodom, a da će rezultat biti isti.

Kad se izrađuju umjetne referentne funkcije, namjerno se izrađuju na određeni način, tako da ističu određene aspekte algoritma. Problemi s referentnim vrijednostima su umjetni i ljudi koji razumiju kako ovi algoritmi rade promjenom ovog ili onog dijela ispituju ponašanje algoritma. Ali u prirodi, problemi to ne čine. Kad se rade problemi s umjetnim referentnim vrijednostima, trudi se da oni budu najteži. U prirodi toga nema. Kod spomenutog problema s putujućim trgovcem, njegova struktura se može vidjeti i u praksi se ona ne smatra vrlo teškom. Optimalno rješenje se pronalazi bez većih računanja iako postoji ogroman broj mogućnosti i potrebno je 4000 godina na računalu za sva njihova ispitivanja. Međutim, ako se napravi samo jedna pretpostavku s kojom se npr. podijeli zemlja na sjever i jug. Tada umjesto prijašnjih 20 gradova u državi imate 10 gradova na sjeveru i 10 na jugu. Rješenje se pronalazi u trećini sekunde. Potrebno ih je još međusobno povezati, ali za jednu minutu vrlo dobro rješenje je tu. Svi problemi s prirodom su manje ili više takvi. Dakle, ti problemi imaju određenu strukturu, imaju ogroman broj mogućih rješenja, ali nisu posve nepoznati.

Grana računarske znanosti koja se bavi proučavanjem strojeva i programa koje karakterizira inteligentno ponašanje naziva se umjetna inteligencija (eng. artificial intelligence). Pod inteligentnim ponašanjem misli se na ponašanje inteligentih agenata (eng. intelligent agents). Inteligentni agenti su računalni modeli stvarnih objekata pa ih se naziva i apstraktni inteligentni agenti (eng. abstract intelligent agents). Izraz

autonomni inteligentni agent (eng. autonomous intelligent agents) se koristi ako se želi naglasiti njihova neovisnost. U našem fokusu će biti jedan aspekt umjetne inteligencije, planiranje. U njemu inteligentni agent analizira sve opcije koje vode do cilja u trenutnom okruženju te bira najbolji od ponuđenih puteva. Ako u toj analizi koristimo više agenata (eng. multi-agent planning) dolazimo do pojma inteligencije rojeva. Tu se odluke donose na razini zajednice što sam proces planiranja čini nešto kompleksnijim od klasičnog planiranja izoliranih agenata u drugim modelima [93].

Algoritmi inteligencije rojeva su inspirirani prirodnim procesima. Evolucijski algoritmi (eng. evolutionary algorithm) su također inspirirani prirodnim procesima. Ovdje ih spominjemo radi lakšeg snalaženja među kategorizacijom algoritama umjetne inteligencije, a i razumijevanje jednog od ova dva koncepta olakšava usvajanje onog drugog. Budući da su oba algoritma i evolucijski i inteligencije rojeva inspirirani prirodom, kao takvi dijele i mnoge sličnosti. Oba pristupa, u rješavanju problema, sastoje se od idućih koraka:

1. nezavisno pretraživanje prostora problema
2. evaluacija prikupljenih podataka i određivanje najboljeg rezultata
3. pomak svih agenata u domeni pretraživanja prema najboljem rješenju

U evolucijskim algoritmima agenti su modelirani genima nad kojima se izvršavaju operacije reprodukcije, mutacije, rekombinacije i prirodne selekcije. Takvi algoritmi su metaheuristički ili stohastički. Heurističke tehnike koriste aproksimacije rješenja (kada klasične metode ne mogu dati nikakvo rješenje) ili ubrzavaju procese koji bi inače bili prespori za izračunavanje. Metaheuristika proširuje sam koncept heuristike kontroliranjem osnovnih heurističkih algoritama uz pomoć strojnog učenja. Takve su nam metode, za sada, jedina opcija pri izračunavanju NP-teških potpunih problema. Stohastički model definiramo kao onaj koji prepoznaje slučajnu prirodu ulaznih komponenti, odnosno pokus čiji ishod nije unaprijed određen.

Između 70-tih i 80-tih godina Gerardo Beni prvi proučava ponašanja rojeva čestica razmatrajući njihovu primjenu u robotici. Promatra velik broj jednostavnih entiteta koji surađuje pri realiziranju zajedničkog cilja. U prirodi takvo ponašanje vidimo u mravljim kolonijama prilikom potrage za hranom, načinu na koji jata riba izbjegavaju

predatore ili obrani organizma od toksičnih supstanci protiv kojih djeluje imunološki sustav.

Inteligencija rojeva (eng. swarm intelligence, u daljnjem tekstu oznaka SI) inspirirana je kolektivnom inteligencijom. Velike skupine jedinki, inteligentni agenti, ponašaju se tako da donose odluke na osnovi ili lokalnih informacija dobivenih iz okoliša ili međusobne komunikacije s ostalim jedinkama. Njihova komunikacija može biti direktna ili indirektna. Primjer direktne komunikacije imamo kod pčela koje rezultate svoje pretrage dijele s ostatkom košnice, kasnije detaljnije obrađeno u algoritmu pčela. Primjer indirektne komunikacije, putem okoliša, služe se mravi u potrazi za hranom, kasnije detaljnije obrađeno u optimizaciji kolonijom mrava. Ove interakcije na kraju mogu dovesti do pojave kolektivne inteligencije. Mnoga istraživanja su pokazala da velike skupine jedinki imaju sposobnost rješavanja složenih zadataka uz pomoć kolektivne inteligencije tj. inteligencije rojeva [39].

Kako je već rečeno, inteligentni agenti reprezentiraju jednu jedinku u skupini i njeno ponašanje je jednostavno implementirati na računalu. Cijeli sustav inteligencije rojeva bazira se na jednostavnim pravilima. Individualni agenti donose odluke na osnovi lokalnih informacija, nema centralizirane kontrole koja utječe na njihovo ponašanje. Ponašanje agenata je lokalno, u nekoj određenoj mjeri i slučajno, a iz međusobne komunikacije agenata dolazi do pojave samoorganizacije i globalno inteligentnog ponašanja koje nije poznato individualnim agentima. U algoritmima inteligencije rojeva, svaka individua predstavlja rješenje u prostoru pretraživanja. Izazov pri primjeni algoritama inteligencije rojeva leži u ispravnom odabiru prirodnog modela i prilagođavanju zadanog problema zahtjevima algoritma.

Inteligencija rojeva uspješno je našla primjenu u mnogim optimizacijskim i istraživačkim problemima, kao što su kombinatorna optimizacija, optimizacije funkcija, pronalaženje optimalnih putanja, strukturna optimizacija, analiza slika i podataka, itd. Veličina širenja ove tehnologije vidi se i u činjenici da danas pronalazi primjenu i u bioinformatici i medicinskoj informatici (eng.bioinformatics and medical informatics), strojnom učenju (eng. machine learning), dinamičkim sustavima (eng. dynamical systems and operations), itd.

Primjena algoritama inteligencije rojeva ima smisla budući da iterativni proces algoritma nalikuje na samoorganizirajuću evoluciju sustava [54].

Samoorganizacija u rojevima ima sljedeće četiri karakteristike [19]:

1. Pozitivne povratne informacije: služe poticanju na kreiranje prikladnih struktura. Primjer pozitivnih povratnih informacija su mravi koji ostavljaju feromonski trag;

2. Negativne povratne informacije: služe za stabilizaciju populacije. Potrebne su kako ne bi došlo do zasićenja npr. u broju dostupnih tragalaca za hranom (foragers);

3. Fluktuacije: slučajnosti u kretanju i obavljanju zadataka. Od vitalnog su značaja za očuvanje kreativnosti jer omogućuju otkrivanje novih rješenja;

4. Višestruke interakcije: svaka individua koristi, šalje i prima informacije od ostalih članova roja (jata), i na taj način se informacije šire kroz roj.

Kao i svaki drugi algoritam i SI algoritam ima prednosti i nedostatke, odnosno ograničenja koja su postavljena. Prva njegova veća prednost je mogućnost skaliranja sustava, ima mehanizam kontroliranja svih agenata neovisno o tome koriste li se male grupe ili veće grupe od više tisuća subjekata. Iduće ključno obilježje je adaptivnost. Za vrijeme izvođenja, zbog samoorganizirajućih i autokonfigurirajućih svojstava, agenti se mogu dinamički prilagođavati promjenjivoj okolini. Iz samog modela slijedi da individua sama za sebe nije ključna za daljnje izvođenje. To omogućuje cijelom sustavu vrlo visoku sposobnost tolerancije greške, tj. kolektivnu robusnost.

U prirodi ukoliko se jedna ptica odvoji od jata to neće narušiti formaciju cijelog jata, naravno ako je to jato dovoljno veliko. Time je problem jedinstvene točke neuspjeha (eng. single point of failure) zaobiđen. S druge strane, nedostatci SI algoritmima su postavljanje parametara sustava, opasnost od stagnacije i zadaće koje iziskuju brzo rješavanje.

Sve stohastičke optimizacijske metode imaju za izazov podešavanje parametara koji se najčešće naštimavaju metodom pokušaja i pogreške ili ih se adaptira tijekom izvođenja. Budući da je algoritam samo način rješavanja problema, postoji svojevrsno prilagođavanje algoritma na konkretan problem, ispitivanje domene rješenja i njegovo prilagođavanje u skladu s tim.

S porastom broja agenata, ali i s porastom izmijenjenih generacija agenata raste vjerojatnost ispravnog rješenja. Kao posljedica toga slijedi duga obrada podataka

koja uključuje evaluacije okruženja svake jedinke unutar svake generacije, kroz više uzastopnih generacija. Može se dogoditi da će svo to vrijeme računanja biti uzaludno, u slučaju prijevremene konvergencije algoritma k lokalnom optimumu, koji ne mora biti traženi globalni optimum. Uspješan oporavak sustava iz takvih situacija može se postići posebnim mehanizmima nasumičnih odabira čime se unosi raznovrsnost u populaciju. Najčešći uzroci takvih situacija su nedovoljno dobri početni parametri.

4.2 Značajni predstavnici algoritama inteligencije rojeva

Prva dva algortima iz ove grupe algoritama su mravlji algoritmi i algoritam optimizacije rojevima čestica. Dorigo i Colorni su u svom radu napisali mravlji algoritam [33], dok su Kennedy i Eberhart tvorci algoritma optimizacije rojevima čestica [55]. Ubrzo nastaju i sljedeći algoritmi inteligencije rojeva kao što su rojevi pčela, jata riba, grupa kukavica, svitaca i šišmiša, kolonije bakterija, itd. U ovom poglavlju dan je opis najzastupljenijih algoritama i njihovi pseudokodovi.

4.2.1 Algoritam optimizacije kolonijom mrava

Algoritam optimizacije kolonijom mrava (eng. Ant colony optimization algorithms, ACO) prvenstveno je dizajniran za problem trgovačkog putnika i problem bojanja grafa [35]. Algoritam optimizacije kolonijom mrava predložen je od M. Doriga. Inspiracija je dobivena promatranjem ponašanja kolonije mrava kroz okolinu u potrazi za hranom. Točnije, kako mravi pronalaze najkraći put od svoga gnijezda do izvora hrane [34]. Tokom svoga kretanja mravi ostavljaju supstancu koja se naziva feromon. Preko feromona oni indirektno komuniciraju. Svi su mravi jednako jako privučeni feromonima i kreću se u pravcu jačeg feromonskog traga. Jedan primjer ovakvog ponašanja je nalaženje kraćih puteva od mravinjaka do izvora hrane. U početku se mravi kreću dosta kaotično. Međutim kako vrijeme prolazi, sve veći broj mrava se kreće kraćim putem od mravinjaka do izvora hrane, a na kraju skoro svi.

Ovakvo ponašanje je posljedica toga što izbor puta zavisi od količine feromona na određenoj stazi. Feromon isparava. Logično je da je na dužim putevima, na kojima je mravima potrebno više vremena dok dođu do hrane, trag feromona slabiji nego na kraćim. Zbog toga sljedeći put kada mrav bira put, puno je veća vjerojatnost da izabere kraći put. Na kraju, skoro svi mravi koriste kraće puteve jer se na njima nalazi

veća količina feromona. Ovaj indirektni oblik suradnje je poznat kao stigmergija.

U algoritmu 1 za ACO prvo se inicijaliziraju informacije feromona. Algoritam se uglavnom sastoji iz dva iteracijska koraka: konstrukcije rješenja i ažuriranje feromona.

Algorithm 1 Predložak za ACO algoritam

```
Inicijalizacija tragova feromona;
while nije zadovoljen kriterij zaustavljanja do
  for svakoga mrava do
    Konstrukcija rješenja koristeći trag feromona;
  end for
  Ažuriranje tragova feromona:
  Isparavanje;
  Pojačavanje;
end while
Pronalazak najboljeg rješenja ili skupa rješenja.
```

1. Konstrukcija rješenja: Izrada rješenja vrši se prema vjerojatnosnom pravilu tranzicije. Inteligentni agenti, mravi, mogu se smatrati stohastičkom pohlepnom procedurom koja konstruira rješenje na pravilima vjerojatnosti dodavanjem komponenti rješenja u djelomične sve dok se ne dobije cjelovito rješenje. Cilj optimizacijskog problema može se promatrati kao graf odlučivanja (ili graf građenja) gdje će mrav odlučiti o svom putu. Obično ovo iterativni postupak uzima u obzir.

2. Ažuriranje feromona: Ažuriranje feromona provodi se upotrebom generiranja rješenja. Globalno pravilo ažuriranja feromona primjenjuje se u dvije faze:

   - Faza isparavanja u kojoj se trag feromona automatski smanjuje. Svaka vrijednost feromona smanjuje se za fiksni omjer:

$$\tau_{ij} = (1 - \delta)\tau_{ij}, \ \forall ij \in [0, 1] \tag{8}$$

   gdje $\delta \in [0, 1]$ predstavlja brzinu redukcije feromona. Cilj isparavanja je da se za sve mrave izbjegne preuranjena konvergencija prema dobrim rješenjima, a zatim potakne diverzifikaciju u pretraživačkom prostoru (istraživanje).

   - Faza pojačanja u kojoj se trag feromona ažurira u skladu s dobivenim rješenjima i gdje se mogu primijeniti različite strategije.

4.2.2 Algoritam optimizacije rojem čestica

Optimizacija rojem čestica (eng. particle swarm optimisation - PSO) najopćenitiji je od svih SI algoritama i baza je za mnoge varijacije. To je stohastička metoda za optimizaciju razvijena od strane Eberharta i Kennedyja 1995. godine. PSO tehnika se bazira na analogiji oponašanja jata ptica ili riba. Prilikom kretanja za hranom ili u zaštiti od neprijatelja kada jedna od jedinki pronađe povoljan pravac kretanja, druge jedinke u jatu vrlo brzo će početi slijediti tu jedinku, čak i u slučaju da se nalaze na sasvim drugoj strani jata. Zaista, u tim jatima (rojevima) se pojavljuje koordinirano ponašanje jedinki korištenjem lokalnih pokreta bez ikakve središnje kontrole. Navedena analogija se realizira u jednostavnom Boid modelu koji je uveden da bi se bolje razumjela pojava i razvoj, a i osnova je za algoritam optimizacije rojem čestica. U Boid modelu svaka je ptica predstavljena kao točka u Decartesovom koordinatnom sustavu kojoj se dodjeljuje inicijalna pozicija i brzina. U modelu svaka točka teži da ima brzinu kao najbliži susjed. Budući da se vrlo brzo dolazi do jednostavnog modela u kojem sve točke imaju jednaku brzinu, brzinama se u svakoj iteraciji dodava slučajna vrijednost da bi model što bolje opisivao realnu situaciju.

Članovi jata prenose međusobno informacije o dobrim pozicijama jedan drugome i ovisno o njima prilagođavaju svoje pozicije i brzine. Komunikacija se provodi na dva načina:

1. preko globalno najboljeg rješenja ("pozicije") koja je poznata svim članovima jata,

2. preko lokalnih najboljih rješenja koja su poznata u određenom susjedstvu jata.

Kennedy i Eberhart su svoj algoritam nazvali algoritam optimizacije rojem čestica, zato što se u njihovim rješenjima jedinke ptica promatraju kao čestice bez mase i volumena, od interesa su samo njihova brzina i položaj. Ažuriranje pozicija i brzina članova čestica se provodi sljedećim formulama u svakoj iteraciji optimizacijskog postupka [46]:

$$v_x = v_x + 2 * rand * (pbestx - x) + 2 * rand * (gbestx - x) \tag{9}$$

$$x = x + v_x \tag{10}$$

gdje je *rand* – slučajan broj u intervalu $[0, 1]$, *pbest* - prethodno najbolja pozicija i *gbest* - globalno najbolja pozicija u jatu.

Svaka čestica, u algoritmu optimizacije rojem čestica, predstavlja potencijalno rješenje optimizacijskog problema u D-dimenzionalnom prostoru pretrage. Budući da svaka čestica pamti, kako svoju najbolju poziciju, tako i najbolju poziciju u jatu, na osnovi datih informacija, u svakoj iteraciji, česticama se izračunava nova brzina, i time određuje nova pozicija. PSO algoritam, u svakoj iteraciji, česticama mijenja njihovu brzinu tako da se kreću u pravcu prethodno najbolje pozicije (*pbest*) i globalno najbolje pozicije u jatu (*gbest*).

Promjena položaja čestice modelira proces pretrage. Taj proces pretrage određuje se znači pod utjecajem *pbest* (individualni faktor) i *gbest* (socijalni faktor). Ako je utjecaj individualnog faktora veći, onda u pretraživanju dominira proces eksploracije. S druge srane, ako je utjecaj socijalnog faktora veći, onda u pretraživanju uglavnom dominira proces eksploatacije. Na taj način PSO balansira između globalnog i lokalnog pretraživanja, čime se na kraju omogućuje fino pretraživanje prostora potencijalnih rješenja problema.

4.2.3 Algoritam umjetne kolonije pčela

U ovom je dijelu opisan opći algoritam umjetne kolonije pčela (ABC), prilagodljiv problemima s diskretnim pretraživačkim prostorom. Algoritam umjetne kolonije pčela je algoritam inteligencije rojeva predložen od strane Karaboge 2005. godine [53]. To je još jedna metaheuristika inspirirana nekim ponašanjem roja, u ovom slučaju posebno pčelama. Četiri glavne faze određuju algoritam ABC. To su početna faza, faza zaposlenih pčela, faza pčela promatrača i faza pčela izviđača koj e se izvode na sljedeći način:

Algorithm 2 ABC Algoritam

```
1: Inicijalna faza
2: repeat
3:     Faza zaposlenih pčela
4:     Faza pčela promatrača
5:     Faza pčela izviđača
6: until zadovoljen kriterij zaustavljanja
```

A. Početna faza: Tijekom ove faze generira se slučajni skup rješenja unutar datog

prostora za pretraživanje. Budući da se algoritam ABC, kao i svaki drugi algoritam temeljen na populaciji, cijelo vrijeme bavi čitavim skupom ili populacijom rješenja, ova inicijalizacija se mora provesti.

B. Faza zaposlenih pčela: Svako rješenje u skupu radnih rješenja tretira se kao izvor hrane, a na svaki izvor hrane pričvršćena je jedna zaposlena pčela. Svaka zaposlena pčela vrši lokalnu pretragu u susjedstvu svoga izvora hrane i pričvrsti se izvoru hrane bolje kvalitete, ukoliko se takav nađe. To znači da se tijekom ove faze za svako rješenje unutar trenutnog skupa rješenja provodi neko proizvoljno lokalno heurističko pretraživanje s pohlepnim izborom. Međutim, preporučuje se da se lokalno pretraživanje koristi kao slučajna pretraga, što znači da je susjedno rješenje nasumično odabrano iz raspoloživog prostora za pretraživanje. Budući da je ova jednostavna metoda pretraživanja s pohlepnim odabirom od značajne važnosti, može se tretirati kao zasebna metoda. Naime, za dano rješenje S njegova je okolina ograničena na proizvoljnu veličinu i iz nje se pronalazi slučajno rješenje S'. Ako je rješenje S' bolje kvalitete, S zamjenjuje S'.

C. Faza pčela promatrača: Za svaku zaposlenu pčelu postoji jedna povezana pčela promatrač. S određenom vjerojatnošću, obično se izračunava na sljedeći način:

$$P_i = \frac{fitness_i}{\sum_j fitness_j} \tag{11}$$

gdje je $fitness_i$ kvaliteta i-tog rješenja, svaka pčela promatrač izvodi još jednu lokalnu pretragu oko izvora hrane pridružene zaposlene pčele i prati novi pronađeni izvor hrane s boljom kvalitetom, ili ako nema izvora hrane s boljom pronađenom kvalitetom prati isti izvor hrane pridružene zaposlene pčele. Tijekom ove faze, za svako rješenje S u trenutnom skupu rješenja provodi se još jedno lokalno pretraživanje njegovog susjedstva, na isti način kao u prethodnoj fazi, ali s vjerojatnošću P_i iz jednadžbe 11. Trag se čuva od novog pronađenog rješenja s boljom kvalitetom, ili ako se ne nađe rješenje s boljom kvalitetom, od rješenja S koje je sada pohranjeno u drugom setu rješenja, koji se razlikuje od skupa zaposlenih pčela. S obzirom na veličinu naselja, posebno kad se vrši lokalna pretraga pčela promatrača, treba osigurati mogućnost da se ona prilično širi. To se može postići spuštanjem granice susjedstva tijekom obavljanja istraživanja iz spomenute metode u fazi pčela promatrača.

D. Faza pčela izviđača: U fazi pčela izviđača svaka zaposlena pčela napušta

svoj izvor hrane ako ga nije mijenjala određeno vrijeme i traži drugi besplatni izvor hrane. To znači da ako se neko rješenje zaposlene pčele nije promijenilo za neki ograničavajući broj iteracija, ono se odmah zamjenjuje s potpuno novim slučajnim rješenjem. Tijekom cijelog postupka pohranjuje se i prati jedno rješenje najbolje kvalitete.

4.2.4 Algoritam krijesnica

U svijetu postoji nekoliko tisuća vrsta krijesnica. Krijesnice su kukci koji mogu međusobno komunicirati pomoću svjetlosnih signala određene jačine i ritma, pri čemu karakteristike svjetlosti ovise od određene vrste krijesnica. Ta sposobnost proizvodnje svjetlosti naziva se bioluminiscencija. Dvije osnovne funkcije svjetlosti koju emitiraju krijesnice su da privuču potencijalnog partnera na razmnožavanje ili potencijalni plijen. Također, svjetlucanje može biti znak upozorenja i odbijanja drugih vrsta koje bi im mogle naškoditi. Privlačnost između dvije krijesnice je veća što je veći intenzitet svjetlosti koje krijesnice odašilju jedna drugoj. Smanjenjem razmaka između njih povećava se intenzitet svjetlosti kojeg vide, a time se automatski povećava i privlačnost među njima. Rezultat toga je pomicanje krijesnica jedne prema drugoj.

Svjetlost koju krijesnice emitiraju i proces privlačenja između njih je inspiriralo Xin-Shi Yanga [134] da razvije algoritam za optimizaciju koji je nazvan algoritam krijesnice (eng. firefly algorithm, FA). U ovom algoritmu emitiranje svjetlosti se formulira na pogodan način i povezana je s funkcijom cilja koju treba optimizirati FA koristi tri pravila koja su idealizirana:

- Krijesnice su bez spola čime se osigurava njihovo međusobno privlačenje bez obzira na spol.
- Privlačnost je proporcionalna intenzitetu svjetlosti koja se smanjuje s povećanjem udaljenosti. Vrijedi, manje svjetla krijesnica kreće se prema svjetlijoj krijesnici, međutim ako nema svjetlije od određene krijesnice, ona će se kretati slučajnom putanjom.
- Intenzitet svjetlosti krijesnice određen je funkcijom cilja.

Osnovni koraci realizacije FA algoritma su dati pseudokodom u Algoritmu 3 [140].

Algorithm 3 Algoritam krijesnice

```
Funkcija cilja f(x), x = (x_1, x_2, ..., x_d)^T;
Generiranje inicijalne populacije krijesnica, x_i, i = 1, 2, ..., n;
Intenzitet svjetlosti I_i krijesnice x_i je određena preko f(x_i);
Definiranje koeficijenta apsorpcije svjetlosti, γ;
while t < MaksGeneracija do
   for i = 1 to n do
      for j = 1 to i do
         if I_i < I_j then
            pomicanje krijesnice i prema krijesnici j
         end if
         Varirati privlačnost sa distancom pomoću exp[−γr];
         Evaluiranje novih rješenja i ažuriranje intenziteta svjetlosti;
      end for
   end for
   Rangiranje krijesnica i pronalaženje trenutno globalno najbolje rešenje (global best)
   g;
end while
Vizualizacija;
```

Ako intenzitet emitirane svjetlosti krijesnice opada s kvadratom udaljenosti r od izvora i γ je koeficijent apsorpcije svjetlosti tada je formula za izračunavanje intenziteta svjetlosti [135]:

$$I = I_0 e^{-\gamma r} \tag{12}$$

gdje je I_0 originalni intenzitet određen funkcijom cilja.

Budući da je privlačnost krijesnica proporcionalna intenzitetu svjetlosti koju vide susjedne krijesnice, možemo definirati razinu privlačnosti β ovisnu o udaljenosti r sa:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{13}$$

gdje je β_0 privlačnost u točki $r = 0$, a β_0 na početku definiramo s vrijednosti 1.

Udaljenost između bilo koje dvije krijesnice i i j, na pozicijama x_i i x_j, respektivno, računa se uz pomoć Euklidske udaljenosti:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{14}$$

gdje je $x_{i,k}$ k-ta komponenta prostorne koordinate x_i i-te krijesnice. U slučaju

dvodimenzionalnog sustava, dobiva se:

$$r_{ij} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2} \tag{15}$$

Kretanje krijesnice i koja je privučena od svjetlije ili atraktivnije krijesnice j određeno je sljedećom jednadžbom [137]:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\epsilon_i. \tag{16}$$

Drugi pribrojnik određuje utjecaj privlačnosti krijesnice j. Parametar α je parametar slučajnosti pa time utječe na različitost rješenja. Tokom izvršavanja algoritma njegova se vrijednost može mijenjati. Parametar α se definira

$$\alpha_t = \alpha_0 \delta^t \tag{17}$$

gdje je α_0 slučajno odabrani broj iz intervala $[0, 1]$, a δ je faktor hlađenja iz intervala $[0.95, 0.97]$ [134]. Vektor slučajnih brojeva ϵ_i je vektor s vrijednostima na intervalu $[0, 1]$.

Parametar γ je veoma važan u određivanju brzine konvergencije FA algoritma. Pokazalo se da ako vrijedi $\gamma = 0$, algoritam se reducira na jedan oblik algoritma optimizacije rojem čestica (engl. particle swarm optimization). Obično je vrijednost ovog parametra u intervalu $[0.1, 10]$. Što se tiče veličine početne populacije, najčešće se određuje kao n = 25 do 40.

4.2.5 Algoritam šišmiša

Algoritam šišmiša (eng. bat algorithm, BA) također je algoritam inteligencije rojeva predložen od Xin-She Yang [136]. Algoritam je nadahnut eholokacijskim ponašanjem mikro-šišmiša. Pri letenju i lovu, šišmiši emitiraju kratke i ultrazvučne impulse u okoliš i analiziraju njegov odjek, eho. Istraživanja pokazuju da šišmiši uz pomoć eho informacija mogu stvoriti preciznu sliku svoje okoline i precizno određuju udaljenost, oblike i lokaciju plijena. Sposobnost takvog eholociranja mikro-šišmiša je fascinantna, jer ta vrsta šišmiša može pronaći svoj plijen i ustanoviti razlike među različitim vrstama insekata čak i u potpunoj tami [139].

Promatrajući ponašanje i karakteristike mikro-šišmiša, Yang [139] je predložio standardni BA. U algoritmu se koriste idealizirane karakteristike eholokacije koje se mogu izraziti pomoću sljedeća tri pravila:

- Svi šišmiši koriste eholokaciju kako bi osjetili udaljenost i poziciju šišmiša x_i^t je rješenje optimizacijskog problema koji se promatra [139].
- Slijepi miševi, kada su u potrazi za hranom, lete brzinom v_i, na poziciji x_i, emitirajući zvuk fiksne frekvencije f_{min}, promjenljive valne dužine λ i jačine A^0. Mogu automatski prilagoditi njihove valne duljine (ili frekvencije) emitiranih zvučnih signala kao i prilagoditi brzinu emitiranja signala $r \in [0, 1]$, zavisno blizini cilja.
- Pretpostavlja se da jačina zvučnih signala može varirati između velike (pozitivne) vrijednosti A^0 do minimalne vrijednosti A_{min}.

Na početku BA potrebno je inicijalizirati populaciju šišmiša. Svaki šišmiš ima svoje znanje koje dijeli s drugim šišmišima iz populacije. Također, svaki šišmiš koristi znanje drugih šišmiša. Svaki je član populacije predstavljen svojom pozicijom x_i^t, brzinom v_i^t, frekvencijom f_i^t i glasnoćom A_i^t u D - dimenzionalnom prostoru pretrage. Inicijalno se parametri postavljaju na slučajne vrijednosti. Nakon inicijalizacije, novi parametri se definiraju na temelju prethodnih. Pravila za ažuriranje položaja i brzine virtualnog šišmiša su sljedeća [136]:

$$f_i = f_{min} + (f_{max} - f_{min}\beta) \tag{18}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) f_i \tag{19}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{20}$$

gdje je β slučajan vektor s vrijednostima na intervalu $[0, 1]$ a x^* je trenutno globalno najbolje rješenje koje je nađeno nakon usporedbe svih rješenja između svih n šišmiša. Novo rješenje za svakog šišmiša stvara se lokalno koristeći slučajni hod (engl. random walk) prema jednadžbi

$$x_{new} = x_* + \epsilon A_t \tag{21}$$

gdje je ϵ slučajan broj u intervalu $[-1, 1]$ i A_t prosječna glasnoća svih šišmiša u zadanoj iteraciji.

Korištenjem sljedećih jednadžbi ažuriraju se glasnoća i brzina signala

$$A_i^t = \alpha A_i^{t-1} \tag{22}$$

$$r_i^t = r_i^0(1 - e^{-\gamma t}) \tag{23}$$

gdje je α konstanta u intervalu $[0, 1]$, a γ konstanta veća od 0. U zavisnosti od optimizacijskog problema kojeg treba riješiti, različite vrijednosti parametara α i γ mogu utjecati na kvalitetu konačnog rješenja

Pseudokod BA algoritma je prikazan u Algoritmu 4 [26].

Algorithm 4 Pseudokod algoritma šišmiša

1: Definiranje funkcije cilja $f(x)$, $x = (x_1, x_2, \ldots, x_d)^T$;
2: Generiranje inicijalne populacije šišmiša, x_i,$i = 1, 2, \ldots, n$, i brzine v_i;
3: Definiranje frekvencije zvučnih signala f_i na poziciji x_i;
4: Inicijalizacija brzine emitiranja signala r_i, i glasnoće A_i;
5: **while** $t <$ *Max broj iteracija* **do**
6: Generiranje novih rješenja podešavanjem frekvencije i ažuriranjem brzina i lokacija/rješenja (jednadžbe (18)-(20));
7: **if** $rand < r_i$ **then**
8: Pronalaženje rješenja između svih najboljih rješenja;
9: Generiranje lokalnog rješenja u okolini izabranog najboljeg rješenja;
10: **end if**Generiranje novog rješenja postupkom slučajnog hoda;
11: **if** $(rand < A_i$ & $f(x_i) < f(x^*))$ **then**
12: Prihvaćanje novog rješenja;
13: Uvećavanje r_i i smanjivanje A_i;
14: **end if**
15: Rangiranje šišmiša i pronalaženje trenutnog najboljeg rješenja x^*;
16: **end while**
17: Vizualizacija;

4.2.6 Brain storm optimizacijski algoritam

Brain storm optimizacijski algoritam (engl. brain storm optimization, BSO) predložio je u svojim radovima Yuhui Shi 2011. godini [98]. Otada se BSO koristio u brojnim primjenama poput izbora regresijskog parametra za podršku vektora [127], optimizacije energije u sistemskim mrežama [13], aktiviranja bežičnog senzora [24] i mnogih drugih. Zbog široke upotrebe BSO-a predložene su različite modifikacije

[21, 25] i hibridizacije [49, 50] BSO-a. Algoritam inteligencije rojeva BSO bazira se na kolektivnom ponašanju ljudskih bića prilikom rješavanja problema. Iz samoga imena algoritma, vidi se da je na neki način povezan s procesom brainstorminga, tj. načinom dobivanja ideja.

Brainstorming metoda ima dvije važne i izrazito odvojene faze: fazu formiranja ideje (divergentnu fazu) i fazu kritičkog razmišljanja (konvergentna faza). U divergentnoj fazi proizvode se originalne ideje za rješenje problema. U fazi kritičkog razmišljanja te se ideje kritički vrednuju i prilagođavaju potrebnim kriterijima. Za generiranje ideja koriste se Ozbornova pravila koja kažu da su sve ideje dobre i da ih treba podijeliti s ostalim ljudima. Pri tome je zabranjeno obezvrjeđivanje ideja i njihovo kritiziranje jer ih se želi što više generirati. Prestankom dolaska novih ideja završava prva faza i prelazi se u drugu, kritičku fazu. U njoj se evaluiraju i implementiraju najbolja rješenja dok se ne dođe do zadovoljavajućeg rješenja.

Na temelju prethodno definiranog brainstorming procesa i pravila koja treba poštovati, interpretiran je i napisan algoritam BSO s nekoliko koraka. Ovi koraci pojednostavljuju cijeli proces, međutim to ne umanjuje elemente za implementaciju ovog optimizacijskog algoritma koji ima i eksploraciju i eksploataciju. Dva glavna operatora u algoritmu su divergentni i konvergentni operator. Rekurzivnom primjenom divergencije i konvergencije u prostoru pretraživanja dolazi se do optimalnih rješenja. Divergencija se odnosi na eksploraciju novih prostora pretraživanja, dok konvergencija predstavlja eksploataciju postojećih područja u kojima se mogu pronaći dobra rješenja. Brain storm algoritam pripada grupi razvojnih algoritama optimizacije rojeva i ima dvije glavne mogućnosti: učenje o sposobnostima (divergentni operator) i razvoj kapaciteta (konvergentni operator). BSO algoritam ima sposobnost grupiranja rješenja.

Ideje su agenti u brain storm optimizacijskom algoritmu i predstavljene su kao $d-$ dimenzionalni vektori. Proces brainstorminga počinje generiranjem inicijalnih ideja tj. implementira se generiranje n slučajnih rješenja ($d-$dimenzionalnih vektora). Te inicijalne ideje su grupirane u $m-$ grupa, obično se to radi algoritmom k-srednjih vrijednosti (engl. k-mean algorithm). U svakoj grupi izabrana je najbolja ideja, odnosno rješenje s najboljom vrijednošću funkcije cilja, koja se postavlja za centar grupe.

U iterativnom procesu kombinacijama postojećih ideja tj. rješenja kreiraju se nova rješenja. S vjerojatnošću p_{6b} odabire se jedno od rješenja $X^d_{selected}$ koje će potencijalno biti zamijenjeno novim rješenjem. Nakon uspoređivanja starog i novog rješenja, čuva se ono rješenje koje ima bolju vrijednost funkcije cilja.

Nove ideje generiraju se prema slijedećoj jednadžbi:

$$X^d_{new} = X^d_{selected} + \xi * n(\mu, \sigma) \tag{24}$$

gdje je X^d_{new} – d-ta dimenzija novokreirane individue, $X^d_{selected}$ – d-ta dimenzija individue koja je izabrana radi generiranja nove individue, $n(\mu, \sigma)$ – funkcija Gaussove raspodjele; a ξ – težinski koeficijent kojim se određuje doprinos Gaussove slučajne vrijednosti. Ovaj koeficijent se naziva još i veličina koraka (step - size).

Opisani koraci brain storm optimizacijskog algoritma predstavljeni su u algoritmu 5.

Algorithm 5 Pseudokod brain storm optimizacijskog algoritma

Inicijalizacija
Generiranje n potencijalnih rješenja metodom slučajnog izbora.
repeat
  Grupirati n rješenja u m grupa metodom slučajnog izbora.
  Rangirati rješenja unutar grupe i postaviti najbolje kao centar grupe.
  Generirati slučajan broj r između 0 i 1 metodom slučajnog izbora.
  if $r < p_{5a}$ **then**
    Izabrati jedan centar grupe na slučajan način.
    Generirati slučajno rješenje i postavi ga umjesto već odabranog centra grupe.
  end if
  repeat
    Generirati novo rješenje.
    Generirati slučajan broj r između 0 i 1.
    if $r < p_{6b}$ **then**
      Izaberi grupu s vjerojatnošću p_{6bi}.
      Generirati slučajan broj r_1 između 0 i 1.
      if $r_1 < p_{6bii}$ **then**
        Dodati slučajnu vrijednost na centar grupe kako bi se generiralo novo rješenje.
      else
        Izabrati slučajno rješenje iz odabrane grupe i dodati slučajnu vrijednost na njega kako bi se generiralo novo rješenje.
      end if
    else
      Izabrati metodom slučajnog izbora dvije grupe.
      Generirati slučajan broj r_2 između 0 i 1
      if $r_2 < p_{6c}$ **then**
        Kombinirati centre grupa kako bi se generiralo novo rješenje.
      else
        Izabrati po jedno rješenje iz svake odabrane grupe na slučajan način i generirati novo rješenje kao njihovu kombinaciju.
      end if
    end if
    Sačuvati generirana rešenja u prethodnim koracima ukoliko su bolja od prethodnih.
  until n novih rešenja je generirano.
until Kriterij zaustavljanja je zadovoljen.

U pseudo kodu navedenog u algoritmu 5 uočavamo nekoliko parametara koje je potrebno definirati. Parametri p_{5a}, p_{6b}, p_{6bi}, p_{6bii} i p_{6c} određuju kako će biti generirano novo rješenje. Parametar p_{5a} kontrolira frekvenciju promjene centra grupe novim slučajnim rješenjem, odnosno eksploraciju, p_{6b} se odnosi na vjerojatnost da se novo rješenje generira na osnovi jednog ili dva prethodna rješenja, p_{6bi} je vjerojatnost izbora grupe gdje se bira rješenje koje će se koristiti za generiranje novog rješenja

i računa se kao broj rješenja u grupi podijeljen ukupnim brojem rješenja odnosno veličinom populacije. Parametar p_{6bii} se odnosi na vjerojatnost izbora centra grupe ili slučajnog rješenja iz grupe. Na kraju, uz pomoć vjerojatnosti p_{6c} se utvrđuje da li se kombiniraju dva centra grupe ili dvije slučajne ideje iz dvije grupe.

4.2.7 Algoritam krdo slonova

Algoritam za optimizaciju krda slonova (EHO) je nedavni algoritam inteligencije rojeva, a kreirali su ga Wang et. dr. u 2016. [121]. Inspiracija iz prirode za ovaj algoritam je ponašanje krda slonova. Njihovo ponašanje pojednostavljeno je za potrebe algoritma, jer je u prirodi vrlo složeno, a i nije potrebno ugraditi sve elemente. Ponašanje slonova svedeno je na minimum kako bi se mogla provoditi eksploracija i eksploatacija. Ponašanje slonova opisano je na sljedeći način. Populacija slonova dijeli se na određeni broj klanova. U svakom je klanu zakon matrijarhata - svi slijede glavnu ženku u klanu. Osim toga, u svakoj generaciji određeni broj muških slonova napušta klan i živi usamljenim životom. Kad se ovo ponašanje transformira u algoritam matrijarhat u jednom klanu, primjer je koji ima najnižu vrijednost fitness funkcije (za minimiziranje), a muški slonovi koji idu izvan klana su najgora rješenja. To se koristi za osiguravanje eksploracije. U početku se rješenja generiraju nasumično.

Formalni opis EHO-a je sljedeći. Populacija slonova podijeljena je u k klanova. Svaki klan vodi matrijarh. Svaki član j iz klana i pomiče se s obzirom na matrijarh c_i [121]:

$$x_{new,ci,j} = x_{ci,j} + \alpha \left(x_{best,ci} - x_{ci,j}\right) \times r \tag{25}$$

gdje $x_{new,ci,j}$ predstavlja novi položaj slona j u klanu i, dok $x_{ci,j}$ predstavlja njegov stari (trenutni) položaj, $x_{best,ci}$ je najbolje rješenje klana c_i, $\alpha \in [0,1]$ je parametar algoritma, konstanta koja određuje utjecaj matrijarhata, dok je $r \in [0,1]$ je slučajni broj koji se koristi za potencijalno poboljšanje raznolikosti populacije u kasnijim fazama algoritma. Položaj matrijarhata, tj. najbolji slon u klanu ažurira se sljedećom jednadžbom [121]:

$$x_{new,ci} = \beta \times x_{center,ci} \tag{26}$$

gdje $\beta \in [0,1]$ je drugi parametar algoritma koji kontrolira utjecaj od $x_{center,ci}$ a definiran je:

$$x_{center,ci,d} = \frac{1}{n_{ci}} \times \sum_{l=1}^{n_{ci}} x_{ci,l,d} \tag{27}$$

gdje $1 \leq d \leq D$ je d^{ta} dimenzija gdje je D ukupna dimenzija prostora i n_{ci} je broj slonova u klanu i.

U svakoj generaciji i u svakom klanu n_{ci} broj slonova se pomiče živjeti daleko od klana. Slonovi s najgorim fitnes vrijednostima odabiru se za pomicanje i njihov novi položaj izračunava se sljedećom jednadžbom:

$$x_{worst,ci} = x_{min} + (x_{max} - x_{min} + 1) \times rand \tag{28}$$

gdje x_{min} i x_{max} predstavljaju donju i gornju granicu prostora za pretraživanje, respektivno. Parametar $rand \in [0,1]$ predstavlja slučajan broj iz jednolike raspodjele.

Pseudokod cjelovitog algoritma za optimizaciju stada slonova prikazan je u Algoritamu 6.

Algorithm 6 Pseudo kod EHO algoritma

Inicijalizacija
Postaviti brojač generacije t=*1*, postaviti maksimum generacije $MaxGen$
Inicijalizirati populaciju
repeat
  Sortirati sve slonove prema njihovoj fitnes funkciji
  for all klanove c_i u populaciji **do**
    for all slonove j u klanovima c_i **do**
      Ažurirati $x_{ci,j}$ i generirati $x_{new,ci,j}$ s Jed. (25)
      if $x_{ci,j}$=$x_{best,ci}$ **then**
        Ažurirati $x_{ci,j}$ i generirati $x_{new,ci,j}$ s Jed. (26)
      end if
    end for
  end for
  for all klanove c_i u populaciji **do**
    Zamijeniti najgoreg slona u klanu c_i s Jed. (28)
  end for
  Evaluirati populaciju po novo ažuriranim pozicijama
until $t < MaxGen$
return najbolje pronađeno rješenje

Prema istraživanju literature, algoritam EHO primijenjen je na višerazinski prag slike [106], standardni set test funkcija CEC 2013 [116], podješavanje PID kontrolera [94] i mnoge druge probleme.

4.2.8 Algoritam vatrometa

Algoritam vatrometa (engl. fireworks algorithm, FWA) su prvi put predstavili 2010. godine Tan i Zhu [102]. Inspiriran je eksplozijama vatrometa noću. Izvorni FWA promatra dvije vrste vatrometa, one koje su dobro proizvedene i one loše. Dobro proizveden vatromet generira brojne iskre centralizirane oko centra za eksploziju, dok loše izrađeni vatromet proizvodi samo nekoliko iskri razbacanih po prostoru [102]. Ove dvije vrste vatrometa korištene su za provođenje eksploatacije i eksploracije.

Tijekom posljednjih nekoliko godina algoritam vatrometa primijenjen je na brojne aplikacije u stvarnom svijetu, kao što je podešavanje SVM parametara u [108], detekcija neželjene pošte [47], registracija slike [110], optimizacija portfelja [17], RFID problem s mrežnim planiranjem [114], segmentacija slike [113], itd.

Izvorni FWA ima tri glavna operatera: eksploziju, mutaciju i odabir. U svakoj generaciji svi vatrometi generiraju određeni broj iskre, više ili manje, bliže ili dalje od vatrometa, ovisno o vrsti vatrometa. Gaussova mutacija koristi se za mutiranje položaja vatrometa kako bi se generirale mutirane iskre i osigurala raznolikost rješenja. Odabir se odnosi na biranje rješenja koje će sljedeća generacija spremiti kao poziciju vatrometa. Osim ova tri glavna operatora, implementiran je i operater za mapiranje koji osigurava dopustiva rješenja unutar prostora za pretraživanje. Izvorni FWA iz 2010. godine ima bolje rezultate nego algoritam optimizacije rojeva čestica za većinu jednokriterijskih optimizacijskih benchmark funkcija.

S druge strane, daljnja analiza otkrila je neke nedostatke algoritma, kao što su visoki računski troškovi po iteraciji, problemi ako optimalna točka nije nula, izračun loše amplitude eksplozije itd. Kako bi se ovi problemi odbacili, 2013. Zheng, 2013. Janecek i Tan, predložili su algoritam poboljšanog vatrometa (EFWA) [146]. EFWA uvodi pet glavnih poboljšanja FWA-e. Prva izmjena bila je poboljšanje vrijednosti amplitude eksplozije. Problem je bio u tome što bi, ako je vrijednost fitnes funkcije jednaka ili blizu nuli, amplituda eksplozije bila vrlo mala, tako da bi iskre bile gotovo u istom položaju kao i vatromet. To je rezultiralo lošom

eksploatacijom. Izmijenjen je izračun amplitude eksplozije. Kako bi se prilagodio FWA promjenjivim fitnes funkcijama, modificirani su generator iskra i operator mapiranja. Veliko vrijeme računanja po iteraciji smanjeno je uklanjanjem operatora odabira na temelju udaljenosti od vatrometa. Umjesto prethodnog operatora odabira, uveden je elitni slučajni odabir što je rezultiralo smanjenjem vremena računanja. Algoritam vatrometa stalno se testirao na različitim problemima optimizacije i svi su pronađeni nedostatci uzeti u obzir od strane autora. To je rezultiralo s više ažuriranih verzija. Treća verzija bio je algoritam dinamičnog vatrometa za pretraživanje (dynFWA) koji su 2014. godine predložili Zheng, Janecek, Li i Tan. Bila je to vrhunska verzija do nedavnog ažuriranja [147]. Empirijska analiza pokazala je da mutacijski operator nije toliko koristan pa je u dynFWA uklonjen. Druga razlika između EFWA i dynFWA bila je dinamička kontrola amplitude eksplozije za najbolji vatromet u populaciji (nazvan jezgre vatromet, CF). Jedna iteracija dynFWA sastoji se od samo dva koraka: eksplozija i selekcija.

Iste godine kao dynFWA, 2014., Li, Zheng i Tan [64] predstavili su algoritam adaptivnog vatrometa (AFWA). AFWA predstavlja nadogradnju EFWA-e gdje je promijenjena samo formula amplitude. Amplituda eksplozije zamijenjena je adaptivnom amplitudnom metodom koja izračunava amplitudu na temelju procijenjenih fitnes funkcija od individue.

U 2015., Yu i Tan predložili su novi operator mutacije koji je isključen u prethodnim verzijama. Predložen je algoritam vatrometa s kovarijantnom mutacijom (FWACM) [63], uveden je operator mutacije koji koristi informacije iskre s boljim vrijednostima fitnes funkcije za generiranje novih iskri. Predloženi FWACM nadmašio je prethodne verzije, dynFWA i AFWA.

Još jednu poboljšanu verziju, okvir suradnje za FWA (CoFFWA) predložili su 2015. Zheng, Li, Janecek i Tan [65]. U ovoj verziji poboljšana je sposobnost istraživanja i iskorištavanja uvođenjem neovisne strategije odabira i strategije izbjegavanja gužve. Umjesto da odabere najbolje iskre među svim varnicama u jednoj generaciji, svaki vatromet samostalno odabire svoje najbolje iskre koje će spremiti za sljedeću generaciju. Ova je strategija drastično poboljšala sposobnost eksploatacije. S druge strane, eksploracija je poboljšana osiguravanjem raznolikosti rješenja. Svi vatrometi smješteni bliže određenoj udaljenosti od CF-a ponovno su inicijalizirani negdje u

prostoru za pretraživanje (dalje od CF-a). To je osiguralo da se vatromet rasipa oko prostora za pretraživanje, a ne da bude koncentriran oko CF-a.

Algoritam vođenih vatrometa (GFWA) jedno je od nedavnih poboljšanja algoritma vatrometa. Predložili su ga Li, Zheng i Tan 2016. godine [66]. U ovoj verziji izrađena je posebna vođena iskra (GS) za svaki vatromet. Na temelju rezultata dobivenih vatrometom izrađen je vodeći vektor (GV) s obećavajućim smjerom i prilagodljivom duljinom. Vođeni vektor je dodatno dodan položaju vatrometa i rezultat je bio elitno rješenje, GS. Posljednju verziju algoritma vatrometa pod nazivom pojednostavljeni (ogoljeni) algoritam vatrometa (eng. bare bones fireworks algorithm BBFWA) predložili su 2018. godine Li i Tan [67]. BBFWA predstavlja pojednostavljenu verziju algoritma vatrometa koji uključuje samo osnovni eksplozivni operator koji je dovoljan za eksploraciju i eksploataciju. Ovo pojednostavljenje predloženo je kako bi se smanjila složenost algoritma i time smanjilo vrijeme izvršenja. Iako je BBFWA prilično nova verzija, na temelju rezultata predstavljenih u [67], može se smatrati novom modernom verzijom.

U BBFWA-i je broj vatrometa smanjen na jedan, a stvarni agenti su stalni broj iskri. Na ovaj način uklonjen je izračun broja iskre. Vatromet čuva najbolje pronađeno rješenje i u svakoj se generaciji u susjedstvu generiraju iskre. Pojednostavljeni je operator odabira a upotrijebljen je pohlepni algoritam gdje je spremljeno samo najbolje rješenje za sljedeću generaciju.

U algoritmu 7, Lb i Ub označavaju donje i gornje granice prostora za pretraživanje. Vektor x predstavlja najbolje rješenje pronađeno u određenoj generaciji, dok je $f(x)$ vrijednost fitnes funkcije za to rješenje. U svakoj generaciji, n iskri (s_i, $i = 1, 2, \ldots, n$) jednoliko se generira oko najboljeg rješenja pronađenog do tada, tj. u hiper-pravokutniku ograničenom sa $x - A$ i $x + A$, gdje A predstavlja parametar algoritma. Parametar A koristi se za ograničavanje prostora za pretraživanje i time su izvršena eksploracija i eksploatacije. Ako je u jednoj generaciji pronađeno bolje rješenje u usporedbi s prethodnim najboljim rješenjem, sprema se novo najbolje rješenje, a to ujedno znači potrebu za većim istraživanjem prostora. To se postiže povećanjem parametra A množenjem s faktorom $C_a > 1$. S druge strane, ako u jednoj generaciji najbolje rješenje ostane isto, potrebno je veće iskorištavanje. To se postiže smanjenjem parametra A množenjem s faktorom $C_r < 0$. Na taj način

prostor za pretraživanje oko najboljeg rješenja će se smanjiti i nova rješenja će se rasporediti bliže njemu.

Pseudokod za BBFWA dan je u algoritmu 7.

Algorithm 7 BBFWA algoritam [67]

Generirati $\sim U(Lb, Ub)$
Evaluirati $f(x)$
$A = Ub - Lb$
repeat
  for $i = 1$ do n **do**
    Generirati $s_i \sim U(x - A, x + A)$
    Primijeniti operator mapiranja s_i
    Evaluirati $f(s_i)$
  end for
  if $\min_{i=1,2,\ldots,n}(f(s_i)) < f(x)$ **then**
    $x = \text{argmin}(f(s_i))$
    $A = C_a A$
  else
    $A = C_r A$
  end if
until Kriterij zaustavljanja nije zadovoljen
return x

5 Prilagođavanje algoritama inteligencije rojeva za različite prostore pretrage

Nekoliko riječi o tome kako se gotovo svi praktični problemi mogu definirati kao problemi optimizacije i to su najčešće teški optimizacijski problemi. U ovom radu je predstavljeno nekoliko takvih problema.

Sadržaj ovog poglavlja je pregled praktične primjene algoritama inteligencije rojeva. Kroz analiziranje razvoja ovih algoritama izložena je trenutna situacija što se tiče primjene na praktične probleme i dati su mogući smjerovi budućih istraživanja. Budući da područje rješavanja problema optimizacije algoritama inteligencije rojeva više nije potpuna novost, postoje istraživanja s tom tematikom kao i znanja i prijedlozi za daljnji razvoj.

Algoritmi inteligencije rojeva su najprije razvijeni u teoriji, da bi se nakon toga testirali na standardnim test funkcijama tzv. benčmark funkcijama (eng. benchmark functions) kako bi se pripremili za praktičnu upotrebu. Posljednja faza je primjena algoritama inteligencije rojeva na praktične probleme iz svakodnevnog života. Primjena ovih algoritama na praktične probleme podrazumijeva da ih prije svega treba prilagoditi, modificirati, kombinirati, hibridizirati i podesiti za različite tipove problema. Takvi postupci ponekad mogu biti relativno lako napravljeni, a ponekad to može biti poprilično složen posao.

U mnogim slučajevima prilagodba algoritama inteligencije rojeva može biti veliki izazov. U ovisnosti od tih izazova mogu nastati i nove verzije algoritma ili njegove prilagodbe.

5.1 Prostor pretrage za realna rješenja

Svaki algoritam inteligencije rojeva ima svoje parametre uz pomoć kojih se može napraviti prilagodba na određene praktične probleme. Za te parametre postoje intervali dozvoljenih vrijednosti i najčešće postoje preporučene veličine za koje algoritam dobro radi odnosno za koje postoji ravnoteža između faze eksploracije i faze eksploatacije. Mijenjanjem vrijednosti parametrima vrše se razna eksperimentiranja i to su upravo najjednostavnije prilagodbe algoritama inteligencije rojeva na praktične probleme. Ovakve prilagodbe su primjenjive na probleme čija su rješenja vektori čiji

su elementi iz skupa realnih brojeva.

Ovakve prilagodbe su najjednostavnije ali primjenjive na praktične probleme. U ovoj disertaciji razmatrane su i analizirane ovakve prilagodbe algoritama inteligencije rojeva na problem optimizacije stroja potpornih vektora i k–sredina algoritma klasterisanja.

5.1.1 Optimizacija stroja potpornih vektora

Strojno učenje predstavlja važno polje u računalnoj znanosti i bavi se algoritmima koji uče na povijesnim podatcima, traži skrivene veze u podatcima i izrađuje model koji se može koristiti za predviđanje rezultata u budućnosti. Neki primjeri problema strojnog učenja su otkrivanje neželjene pošte [145], predviđanje vrijednosti dionica [103], prepoznavanje govora [44], itd. Prepoznaju se tri vrste algoritama strojnog učenja: nadzirano, nenadzirano i pojačano učenje. Algoritmi nadziranog učenja pretpostavljaju da za podatke koji se koriste za obuku (učenje) izlazne vrijednosti su poznate, a algoritmi učenja bez nadzora ne koriste nikakvo znanje o mogućim izlazima, dok se pojačano učenje oslanja na povratne informacije o kvaliteti rješenja. Tri glavne skupine zadataka strojnog učenja su klasifikacija, regresija i grupiranje.

Klasifikacija predstavlja nadgledani problem učenja gdje je cilj stvoriti model koji će novu instancu staviti u jednu od mogućih klasa. Klasifikacija se široko primjenjuje u raznim znanostima kao što su medicina [46], ekonomija [96], astronomija [57], itd. Danas postoje brojni algoritmi klasifikacije, a neki od najčešće korištenih su SVM, umjetne neuronske mreže, k-najbliži susjed, logistička regresija, naivni Bayesov klasifikator, slučajna šuma.

Jedan od primjera teških problema optimizacije u strojnom učenju je pronalaženje metaparametara stroja potpornih vektora (engl. support vector machine, SVM). Metode potpornih vektora su binarni klasifikatori, ali se koriste i za višestruku klasifikaciju. To je moćan klasifikator široko korišten u prošlosti za razne probleme [111], [144].

Dvije različite metode, "jedan-protiv-jednog"(engl. "one-versus-one") i "jedan-protiv-svih"(engl. "one-versus-all"), koriste se za prilagođavanje binarnih klasifikatora na probleme s više od dvije klase. U metodi jedan-protiv-jednog za svaku kombinaciju od dvije klase kreiraju se modeli. Nove instance klasificiraju se po ovim modelima.

Svaki model klasificira instancu, a na kraju je potrebno samo prebrojati koliko puta je ta instanca pripadala svakoj klasi, a zatim odabrati klasu s najvećim brojem pojava. Osim te metode, postoji metoda jedan-protiv-svih, gdje je izrađen model za svaku klasu radi razlikovanja jedne klase od svih ostalih klasa u kombinaciji. Za metodu jedan-protiv-svih za n klasa kreira se n modela. Nakon povlačenja instance kroz sve modele, idealna situacija bi bila da samo jedan model klasificira tu instancu kao člana te klase, dok ga svi ostali modeli klasificiraju kao člana *druge klase*. Ako se takva idealna situacija ne postigne, za rješavanje iste opet se može koristiti nekakav mehanizam za glasanje ili djelomična metoda jedan-protiv-jednoga. U ovom smo radu koristili metodu jedan-protiv-jednoga.

SVM razdvaja instance iz dvije klase pomoću hiperravnine. Sve instance iz jedne klase smještene su na jednoj strani hiperravnine, dok su sve instance iz druge klase smještene na suprotnu stranu. Hiperravnina postoji samo kada se instance iz dvije klase linearno odvajaju. Broj ovih hiperravnina je u generalnom slučaju beskonačan. Ono što SVM treba učiniti je tzv optimalna hiperravnina.

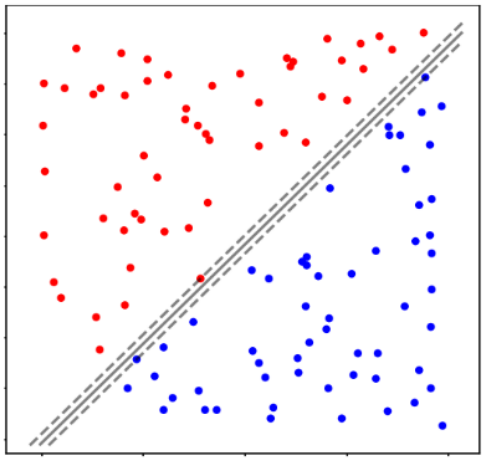
Optimalna hiperravnina je hiperravnina koja je što dalje od instanci obje klase. Ta hiperravnina je postavljena na sredini margine gdje je margina razmak između dvije hiperravnine od kojih je svaka na rubu točaka koje predstavljaju instance. Također, ove dvije ravnine dodiruju najbliže instance između klasa. Formalno, optimalna hiperravnina se definira sljedećom jednadžbom:

$$y_i \ (w \cdot x_i + b) \geq 1 \ \ \textbf{for} \ \ 1 \leq i \leq n, \ w \in R^d, \ b \in R, \tag{29}$$

gdje x_i predstavlja instance, $y_i \in \{-1, 1\}$ su njihove odgovarajuće oznake, b je termin presretanja, a w je normalni vektor za hiperravninu, d označava broj atributa svake instance i n je broj instanci.

Hiperravnina se zapravo definira instancama koje se nalaze najbliže njoj. Ove instance se nazivaju potporni vektori.

Prethodna definicija SVM modela zahtijeva da sve instance koje pripadaju istoj klasi moraju biti na desnoj strani hiperravnine i izvan margine, što je razmak između potpornih vektora (slika 4). Stvarni životni problemi obično nemaju ove karakteristike. Za takve probleme prethodna definicija SVM-a, često zvana tvrda margina (eng. hard margin), nije prikladna. Kako bi se savladao ovaj problem i



Slika 4: *Primjer optimalne hiperravnine i margine*

učinio SVM stvarno upotrebljivim za stvarnu klasifikaciju podataka, predloženo je definiranje mekih margina (eng. soft margin).

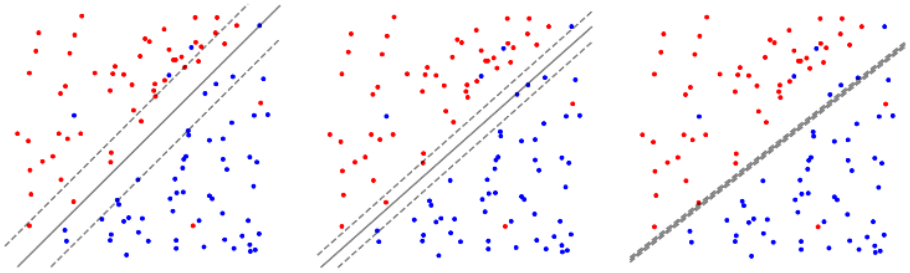
U teoriji, ova definicija SVM-a djeluje u redu, ali instance problema u stvarnom svijetu nisu tako idealno uređene i često ne postoji jaz (margina) između njih. U praksi postoje na primjer ulazne pogreške, iznimke (eng. outliers) odnosno instance značajno različite od ostalih instanci u istoj klasi. Idealni podaci gdje su margina i hiperravnina savršeni vrlo su rijetki, gotovo da i ne postoje. Da bi se savladao ovaj problem, prethodna definicija može biti malo manje stroga. Može se uvesti parametar C koji će dopustiti neke pogrešne klasifikacije, odnosno biće dozvoljeno da pojedine instance budu unutar margine. Više nije potrebno da sve instance iste klase budu na jednoj strani hiperravnine ili da budu izvan margine, tj. dopušteno je imati instance na pogrešnoj strani ili unutar margine. To se postiže promjenom prethodne definicije u sljedeću:

$$y_i(w \cdot x_i + b) \geq 1 - \epsilon_i, \qquad \epsilon_i \geq 0, \qquad 1 \leq i \leq n \tag{30}$$

gdje su ϵ_i varijable koje omogućuju nekim slučajevima da padnu unutar margine. U ovom slučaju pronalaženje optimalne hiperravnine provodi se rješavanjem ekvivalentnog problema, točnije sljedećeg problema kvadratnog programiranja:

$$\min \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\epsilon_i, \tag{31}$$

gdje C predstavlja prethodno spomenuti parametar mekane margine. Parametar C također se naziva parametar središnje margine koji određuje kolika je cijena pogrešne klasifikacije. Ako se postavi da je C vrlo velik, to znači da svaka instanca koja je pogrešno klasificirana drastično povećava vrijednost funkcije troška, pa samim tim se dolazi do toga da će model biti sličan prethodnom, tj. da gotovo sve mora biti izvan margine i na pravoj strani hiperravni. S druge strane, za male vrijednosti parametra C pogrešno klasificirane instance ne mijenjaju značajno funkciju troška, stoga dobiveni model može biti jako loš. Na temelju toga, očito je da parametar C treba pažljivo odrediti da bi se stvorio dobar model. Optimalne hiperravnine i odgovarajuće margine dobivene za različite vrijednosti parametra C su prikazane na slici 5.

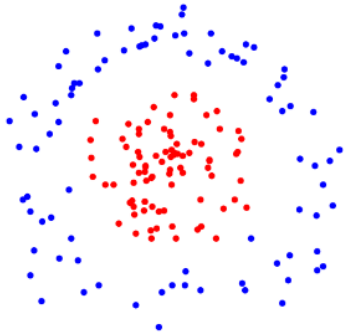


Slika 5: *Primjer optimalne hiperravnine i margine za različite vrijednosti parametra C (C=1 lijevo, C=10 sredina i C=1000 desno)*

Drugi problem s prethodnom definicijom je taj što sve ovisi o linearno odvojivim podatcima u kojima se hiperravnina koja razdvaja instance dvje klase može pronaći. Za podatke, gdje su naprimjer, instance jedne klase unutar kruga, a instance iz druge klase izvan tog kruga, dobra hiperravnina koja razdvaja ove klase ne postoji. Bez obzira na to kako stavimo hiperravninu, jedan broj slučajeva uvijek će biti pogrešno klasificiran. Primjer ovakvog skupa podataka je prikazan na slici 6.

Uobičajeni slučaj u stvarnosti je da podatci nisu linearno odvojivi i zbog toga se koristi jezgreni trik (eng. kernel trick) - umjesto skalaranog proizvoda koji je korišten u prethodnoj definiciji, uvodi se funkcija jezgra. U praksi postoji ograničen broj funkcija koje se koriste kao jezgrene funkcije, a najčešće su polinom, Gaussova radijalna bazna funkcija (RBF) i sigmoidne funkcije. Definicije najčešće korištenih jezgrenih funkcija su dane u tablici 1.

Gaussova radijalna bazna funkcija je najčešće prvi izbor. RBF-jezgra u praksi



Slika 6: *Primjer linearno nerazdvojivog skupa podataka*

Tabela 1: *Najčešće korištene jezgrene funkcije*

Naziv	Definicija	Parametar
Linearni	$k(x,y) = x^T y + c$	c
Polinom	$k(x,y) = (\alpha x^T y + c)^d$	α, c and d
Sigmoid	$k(x,y) = \tanh(\alpha x^T y + c)$	α, c
Power	$k(x,y) = -\|x-y\|^d$	d
Log	$k(x,y) = -\log(\|x-y\|^d + 1)$	d
RBF	$k(x,y) = \exp(-\gamma \, \|x-y\|^2)$	γ

dobro funkcionira i relativno je laka za podešavanje te je prihvaćen kao razumna metoda. Jasno je da ova jezgrena funkcija nije savršena u svim scenarijima, otuda i postojanje drugih, ali obzirom da je RBF-jezgra dobro proučena i široko razumljiva, mnogi SVM paketi je automatski uključuju. RBF-jezgra ima niz drugih svojstava. Jedno od njih je stacionarnost jezgre, što znači invarijantnost za translaciju. Kao stacionarna jezgra davat će istu vrijednost za $K(x,y)$ kao i za $K(x+c, y+c)$, pri čemu je varijabla c vektorska vrijednost dimenzija koja odgovara ulazima. Za razliku od RBF-jezgre, linearna jezgra nema svojstvo stacionarnosti.

RBF-jezgra ima svojstvo da je izotropna, tj. skaliranje po γ se javlja u istoj količini u svim smjerovima.

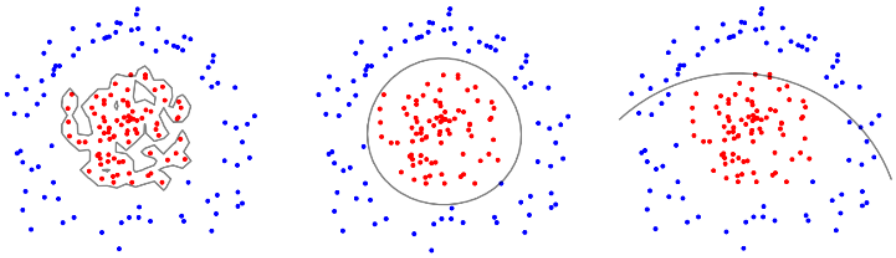
Još jedno svojstvo RBF-jezgre je u tome što je ona beskonačno glatka. To možda nije najvažnije svojstvo, ali je estetski ugodno i pomalo vizualno zadovoljavajuće. Ako se usporede RBF-jezgra i Maternovi kerneli može se vidjeti da postoje neke jezgre koje su prilično nazubljene što otežava podešavanje parametara jezgrenih funkcija.

U ovoj disertaciji, kao i u mnogim primerima u literaturu [27], [70], [23], RBF je

korištena kao jezgrena funkcija. RBF je definirana kao:

$$K(x_i, x_j) = \exp(-\gamma \, ||x_i - x_j||^2), \tag{32}$$

gdje je γ slobodni parametar koji također ima značajan utjecaj na točnost klasifikacije. Ovaj parametar određuje utjecaj na svaku treniranu instancu. Parametar γ određuje utjecaj svake instance na rezultujući model. Male vrijednosti parametra γ znače da utjecaj doseže daleko, dok s druge strane, za visoke vrijednosti γ doseg je prilično blizu. Praktično, vrijednost parametra γ može dovesti to pretreniranog (eng. overfitting) ili nedovoljno istreniranog (eng. underfitting) modela. Na slici 7 je prikazan primjer pronađenih granica s različitim vrijednostima parametra γ (vrijednost parametra C je bio fiksiran na vrijednost 1). Kao što se može vidjeti, za velike vrijednosti parametra γ, granica između klasa je postavljena tako da je apsolutno prilagođena instancama iz trening skupa. Jasno je da ovo predstavlja primjer pretreniranog modela. Na drugoj slici, gdje je $\gamma = 0.01$, granica razdvaja instance savršeno dok ujedno predstavlja i realniji model od prethodnog, te točnost na test skupu može biti veća. U trećem primjeru vrijednost parametra γ je postavljena na 0.00001. Kada je vrijednost parametra γ suviše mala, gubi se smisao uvođenja jezgrene funkcije pa se vraća modelu koji koristi skalarni proizvod.



Slika 7: *Primjer granice dobivene za različite vrijednosti parametra γ (γ=10 lijevo, γ=0.01 sredina i γ=0.00001 desno)*

S obzirom na prethodnu analizu, da bi se napravio optimalni SVM model potrebno je odrediti par parametara C i γ, nije ih dovoljno pojedinačno odrediti. Pronalaženje ovog para je težak optimizacijski problem, jer vrijednosti za C i γ mogu biti bilo koji realni brojevi iz vrlo širokih intervala, tako da postoji beskonačan broj mogućih parova. Točnost funkcije može se značajno razlikovati kada se ovi parametri tek neznatno promijene. Također, točnost modela koji se smatra funkcijom troška ima

ogroman broj lokalnih optimuma, traži se globalni maksimum. Za ovakve probleme uspješno su korišteni algoritmi nadahnuti prirodom, posebno algoritmi inteligencije rojeva. U ovom radu za podešavanje SVM parametara predlažena je upotreba algoritma krdo slonova.

U literaturi je predloženo mnogo različitih pristupa za podešavanje SVM parametara. U [130] predloženo je korištenje razmaka među klasterima u prostorima obilježja za odabir parametara jezgre. Puno manje računalnog vremena se gubi na izračunavanje takve udaljenosti nego za obuku odgovarajućih SVM klasifikatora za pretraživanje mreže, tako da se valjani parametri jezgre mogu odabrati mnogo brže.

U [27] predložen je optimizirani hibridni model umjetne inteligencije radi integriranja brzog neurednog genetskog algoritma sa strojem potpornih vektora. Brzi neuredni genetski algoritam korišten je za podešavanje SVM parametara, dok SVM uglavnom omogućuje učenje i uklapanje krivulje.

Optimizacija rojeva čestica (PSO) široko se koristi za podešavanje SVM parametara. U [70] korišten je jednostavan PSO za podešavanje parametara i odabir obilježja. U [131] PSO koji koriste kaotično mapiranje predložen je za optimizaciju parametara valnog v-potpornog vektora.

PSO algoritam je uveden u model SVM-a najmanjih kvadrata (LS-SVM) kako bi se predložio optimizirani LS-SVM model u [72] s ciljem da se poboljša preciznost klasifikacije u kemijskim podacima. Kaotični algoritam optimizacije korišten je za kaotičnu obradu početnog položaja i lokalnog ekstremnog položaja čestica u algoritmu PSO da bi se dobio kaotični PSO (CPSO). CPSO je korišten za odabir i optimizaciju važnih parametara LS-SVM. Izbjegava se slučajnost odabira parametara i smanjuje se radno opterećenje odabira.

U [23] predložena je paralelna vremenska varijanta algoritma optimizacije rojeva čestica koji istovremeno vrši optimizaciju parametara i odabir obilježja za SVM. U ovoj metodi usvojena je ponderirana funkcija za projektiranje ciljne funkcije PSO-a, koja uzima u obzir prosječne stope točnosti klasifikacije SVM, broj potpornih vektora i odabranih obilježja istovremeno.

Osim PSO-a, za podešavanje parametara korišteni su i drugi algoritmi inteligencije rojeva. U [82] algoritam umjetne kolonije pčela (ABC) korišten je za optimizaciju parametara SVM-a najmanjih kvadrata. Dvije izmjene izvornog ABC-a uvedene

su uz razmatranje kritičnih pitanja poput obogaćivanja strategije pretraživanja i sprečavanje prekomjernog opremanja. Predložena tehnika je uspoređena s dvije tehnike, uključujući neuralnu mrežu širenja unatrag (eng. back propagation) i genetski algoritam.

5.1.2 Primjena algoritma k-sredina na praktične probleme

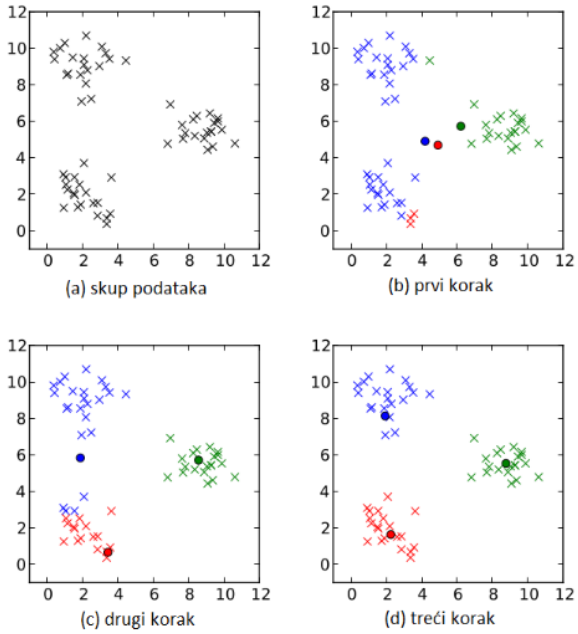
Minimiziranje grupiranja k-sredina je NP-težak problem optimizacije [59].

Algoritam k-sredina definiran je na sljedeći način. Pretpostavimo da skup S sadrži N instanci koje bi trebale biti grupirane u K grupa, $S = X_1, X_2, ..., X_N$. Grupe se određuju njihovim centroidima C_i gdje su $i = 1, 2, ..., K$. Početni centroidi su odabrani nasumično. Iterativni proces započinje nakon inicijalizacije. U svakoj iteraciji instancama se dodjeljuju grupa s najbližim centroidom. Udaljenost se može definirati na različite načine kao Euklidska, Manhattan udaljenost, udaljenost Chebychev, Minkowski udaljenost. Izbor mjere udaljenosti utjecat će na grupiranje, a koja je mjera udaljenosti najprikladnija za koju primjenu ovisi i o skupu podataka i o ciljnoj funkciji. Nakon definiranja grupa, centroidi se ažuriraju. To je drugi korak u svakoj iteraciji. Na temelju instanca koje su dodijeljene jednoj grupi, centroid se izračunava na sljedeći način:

$$c_i = \frac{1}{|S_i|} \sum_{X_j \epsilon S_i} X_j \tag{33}$$

gdje je $|S_i|$ ukupni broj instanci koje su u grupi i gdje je $i = 1, 2, ..., K$. Algoritam k-sredina je gotov s radom kad se dosegnu kriteriji zaustavljanja koji mogu biti: maksimalni broj iteracija, kada centroidi nisu pomaknuti više od konstanta ϵ ili instance ostaju u istoj grupi u dvije uzastopne iteracije.

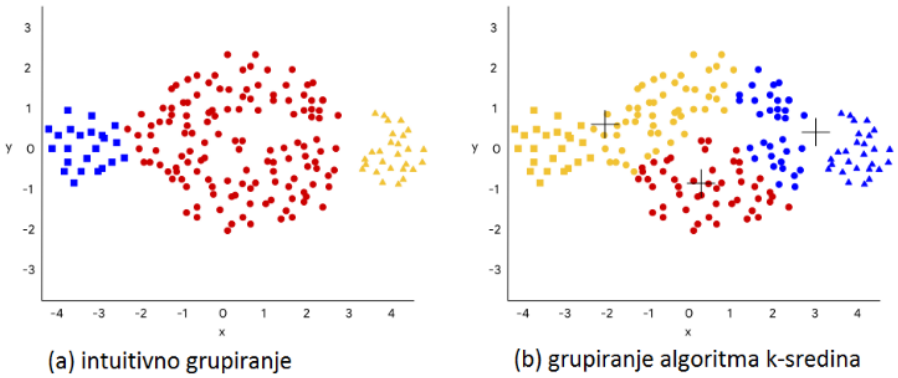
Na slici 8 je dana ilustracija algoritma k-sredina. Skup podataka koji je korišten za ilustraciju je prikazan na prvoj slici a). Ovo predstavlja vrlo jednostavan problem za algoritme grupiranja budući je skup podataka jasno podijeljen na tri različita klastera. Za potrebe ilustracije algoritma k-sredina se za vrijednost parametra k uzima 3, iako u realnosti određivanje broja klastera nije tako jednostavan zadatak. Prva iteracija se sastoji od nasumičnog odabira centara klastera i dodijeljivanja instanci klasteru sa najbližim centrom. Kao što je i očekivano, grupiranje je vrlo niskog kvaliteta to je prikazano na slici b) gdje su različitim bojama označeni nasumično odabrani centri



Slika 8: *Primjer rada algoritma k-sredina*

klastera (prikazani kao kružići različitih boja) kao i dobijeni klasteri (prikazani sa "x" odgovarajuće boje). Može se vidjeti da je jedan klaster (zeleni na slici) uspješno izdvojen iako je centar daleko van tog klastera što će biti regulirano sljedećom iteracijom k-sredina algoritma obzirom da će novi centar postati aritmetička sredina instanci tog klastera. S druge strane, plavi klaster obuhvaća daleko više instanci nego što bi trebalo, dok crveni klaster obuhvaća jedva nekoliko instanci. U drugoj iteraciji, novi centri su preračunaju i ponovlja se postupak grupiranja instanci prema najbližem centru. Na slici c) se može vidjeti da je zeleni klaster sada grupiran oko svog centra. Centar plavog klastera se pomaknuo prema gornjem klasteru i na taj način neke od instanci su postale bliže novom centru crvenog klastera i samim tim promijenile su pripadnost iz plavog u crveni klaster. Konačno u trećoj iteraciji se centri pomiču prema sredinama klastera i ovaj umjetni primjer gdje su klasteri jasno izdvojeni je riješen algoritmom k-sredina. Na ovom primjeru se može lijepo vidjeti napredak algoritma i kako se centri pomiču prema sredinama klastera.

Na slici 9 je dana jedna ilustracija u kojoj se pokazuje usporedba intuitivnog grupiranja i aktualnog grupiranja s algoritmom k-sredina. Primjer pokazuje kako se



Slika 9: *Primjer usporedbe intuitivnog grupiranja i aktualnog grupiranja s algoritmom k-sredina*

algoritam k-sredina može spotaknuti na određene skupove podataka.

Treba imati u vidu da su primjeri iz realnog života mnogo kompleksniji. Prije svega, prethodni primjer je primjer u dvije dimenzije što gotovo nikada nije slučaj sa skupovima podataka dobivenih za praktične probleme. U dvije ili čak tri dimenzije kada se podaci mogu iscrtati i grafički prikazati, algoritmi za grupiranje gotovo da nisu ni potrebni. U slučaju većih dimenzija, problem se značajno komplicira. Nakon prvog koraka, što je određivanje broja klastera, nerijetko treba na tisuće i tisuće iteracija kako bi se centri doveli do sredina klastera. Često se algoritam k-sredina ograničava maksimalnim brojem iteracija budući da uvjet da instance ne promijene klaster odnosno da se centri ne pomiču u dvije uzastopne iteracije dovodi do besmisleno dugog izvršavanja algoritma.

Osnovni algoritam k-sredina definiran je kao problem optimizacije gdje je cilj minimizirati zbroj kvadratne euklidske udaljenosti između instanci unutar svake grupe i njihovih centroida. Fitnes funkcija definirana je na sljedeći način:

$$fit = \sum_{i=1}^{K} \sum_{X_j \epsilon S_i} d(c_i, X_j) \tag{34}$$

gdje $d(c_i, X_j)$ označava kvadrat euklidske udaljenosti između c_i i b.

Grupiranje se uz pomoć algoritama za optimizaciju može izvršiti na sljedeći način. Prostor za pretraživanje definiran je instancama i algoritmom za optimizaciju koji pretražuje k centroida. Dimenzija problema jednaka je $k * d$ gdje je d broj atributa

koji predstavljaju jednu instancu. Ciljna funkcija ima minimizirati zbroj udaljenosti instanci od odgovarajućih centroida data jednadžbom (34).

5.2 Prostor pretrage za cjelobrojna rješenja

Jedna od prilagodbi algoritama inteligencije rojeva odnosi se na prilagođavanje algoritma za cjelobrojne probleme. Za pojedine probleme prilagođavanje algoritma može uključivati samo jednostavno zaokruživanje rješenja na najbliži cijeli broj (problemi iz područja procesiranja digitalnih slika i razni lokacijski problemi) dok u nekim situacijama da bi se algoritam prilagodio rješavanju problema, potrebno je u značajnoj mjeri izmjeniti formule generiranja rješenja, računanja bliskih rješenja, računanje funkcije cilja, i drugo.

Budući da je prostor pretrage kod algoritama inteligencije rojeva u većini slučajeva podskup prostora realnih brojeva, njihove najjednostavnije prilagodbe upravo su one za primjenu na probleme u kojima su rješenja u skupu cijelih brojeva. Slijedi nekoliko primjera najjednostavnijih prilagodba u kojima su vrijednosti rješenja cijeli brojevi.

Najgrublje mogu se izdvojiti dva moguća slučaja ovih prilagodbi, prilagodbe za probleme kada su rješenja cijeli brojevi iz velikog intervala i s druge strane kada su ona iz malog intervala. Ako su oni iz velikog intervala, tada nema većih problema i primjenjuju se algoritmi kao za prostor pretrage za realne brojeve i generirana rješenja zaokružuje se na najbliži cijeli broj. Savršeni primjer za to je obrada slike. U obradi digitalnih slika cijeli brojevi su u intervalu $[0, 2^s - 1]$ gdje je s broj bitova potreban za zapis vrijednosti jednog piksela što je najčešće 8, te je interval pretrage $[0, 255]$ što je veliki interval u smislu razmatranog problema prilagodbe algoritama inteligencije rojeva na probleme sa cjelobrojnim rješenjima. U ovom slučaju rješenja se mogu tražiti primjenom algoritama za realne brojeve i dobiveni rezultati zaokružuju se na najbliži ključni cijeli broj. U ovom konkretnom slučaju, prilagodbe za primjene pri obradi digitalnih slika, ne samo da je prilagodba laka i prihvatljiva, već je i veoma realna jer izvorne vrijednosti piksela predstavljaju izmjeren intenzitet svjetlosti što su realni brojevi koji se zaokružuju na cijeli broj za potrebe čuvanja na računalu. Znači, ta slika se pohranjuje u računalo kao cijeli broj, ali iz prirode su došli kao realni brojevi.

Međutim, u suprotnom slučaju, kada su ti intervali manji, onda dolazi do problema pošto se ovakav vid prilagodme ne može uspješno primjeniti. Jedan od primjera je problem raspoređivanja dronoma za nadzor nad određenim područjem gdje se može optimizirati nekoliko stvari kao što su na primjer položaj svakog drona, smjer leta, broj dronova i slično. Među navedenim promenljivima problema, broj dronova je, očito, cijeli broj. Ako je taj broj mali, recimo tri, četiri ili pet, zaokruživanje generiranih rješenja na cijeli broj kao u prethodnom slučaju ne bi bilo dobro rješenje. U literaturi se mogu naći mnogi primeri gdje se upravo ovo predlaže za cijelobrojne promenljive tražene u malom intervalu odnosno mijenjaju ih kao regularnu varijablu i zaokružuju na najbliži cijeli broj. Ovo može dovesti do nepredvidivog ponašanja algoritma i to se može ilustrovati pomoću nekoliko sljedećih primjera. Polazeći od situacije kada je određeno područje prekriveno s pet dronova i dobiveni su dobri rezultati, premještanjem istih na različite pozicije s ciljem poboljšanja tih rezultata ali činjenica je da dobiveni rezultati mogu biti bolji ili gori: bolji ukoliko se dronovi čiji prostori pokrivanja su se preklapali razdvoje odnosno gori ukliko se dva drona približe tako da im se prostori pokrivanje poklope. U oba slučaja, promjene rješenja u skladu su sa promjenama vrijednosti objektne funkcije. S druge strane, promjenom broja dronova, s njih pet na četiri, dolazi se do dramatične promjene. Pri uklanjanju samo jednog tog drona, područje koje je ranije bilo pod nadzorom postaje otkriveno i vrijednost objektne funkcije se dramatično menja pri minimalnim promjenama rješenja. Ukoliko je na primjer vrijednost promjenljive koja kontrolira broj dronova bila 4.51 te je korišteno 5 dronova, pri samo promeni od 0.02, odnosno pri promjeni vrijednosti promenljive sa 4.51 na 4.49, broj dronova se mjenja na 4, a pokrivenost koja predstavlja objektnu funkciju menja se za ukupno pokrivanje jednog od pet dronova, što je promjena od 20%. Može se zaključiti da ovo rješenje uopće nije dobro jer na ovaj način algoritam postaje slijep i gube se sve prednosti vođenja pretrage.

U ovakvim slučajevima potrebna je složenija tehnika prilagodbe u kojoj će se tražiti rješenja koja su slična, ali ne u prostoru vektora rješenja, već u prostoru objektne funkcije. Da bi algoritmi inteligencije rojeva mogli djelovati, potrebna im je eksploatacija i eksploracija. S eksploatacijom obično nema problema, ali na eksploraciju treba obratiti pozornost i definirati šta su susjedska rješenja. Kada su neka dobra rješenja pronađena, postoji potreba za njihovom kombinacijom s ciljem

pronalaska još boljeg rješenja s tom informacijom. Za to je potrebno imati pojam bliskosti odnosno potrebno je znati kada je jedno rješenje blizu drugog rješenja. Na taj način, dva rješenja koja su bliska mogu se kombinirati i generirati novo rješenja koje im je blizu. U slučaju da su rješenja cijeli brojevi poput tri, četiri ili pet onda rješenja četiri i pet nisu bliska, oni su uistinu daleko jedan od drugog. Tu dolazi do promjene, jer ideja je kombinirati dva rješenja koja su bliska s ciljem dobivanja trećeg sličnog njima.

Stoga, ako postoji rješenje s pet dronova i jednog se povuče, ta dva rješenja više nisu bliska jedano drugome i nisu slične kvalitete. Jedan od načina rješavanja ovog problema je uvođenje parametara koji predstavljaju vjerojatnoću uključiva drona. Ukoliko je najveći broj dronova pet, onda rješenje ima 5 dodatnih realnih parametara čije su vrijednosti u intervalu od 0 do 1. Pokrivenost svakog drona se u funkciji cilja računa se da postoji sa generiranim vjerojatnoćama. Druga opcija je da umesto mjenjanja dimenzije rješenja ili operatora za generiranje novih rješenja bolje je prilagoditi funkciju cilja tako da se rješenja sa četiri i pet dronova budu slične kvalitete, odnosno da predstavljaju bliska rješenja. Način prilagodbe funkcije cilja zavisi od konkretnog problema.

5.2.1 Segmentacija digitalnih slika

Digitalne slike postale su dio brojnih znanstvenih područja. Zbog vrlo moćnih metoda obrade, one se koriste u različitim primjenama, a u posljednjih nekoliko desetljeća predloženi su različiti pristupi analizi digitalne slike. Digitalne metode obrade slike mogu poboljšati analizu slike i koriste se za otkrivanje različitih obilježja koje nisu nužno vidljive ljudskom oku i na taj način olakšavaju veliki napredak u različitim područjima. Pored poboljšanja u analizi, sam proces je obično mnogo brži u usporedbi s ručnom analizom od strane stručnjaka.

Znanstvena područja koja koriste digitalne slike su brojna: medicina, sigurnost, poljoprivreda, astronomija, itd. Medicina predstavlja jedan od primjera koji imaju koristi od napretka u digitalnom snimanju. U medicini, zbog složenosti ljudskog tijela, korištene su digitalne slike iz različitih izvora. Pored digitalnih slika koje hvataju vidljivu svjetlost, postoje i brojni drugi izvori koji se koriste za stvaranje digitalne slike. Neki od poznatih izvora i uređaja koji se koriste za dobivanje medicinskih

digitalnih slika uključuju rentgen, CT skener, magnetsku rezonancu (MRI), PET skener, ultrazvuk i druge. Različiti izvori koriste se za snimanje slika različitih dijelova tijela, različitih tkiva i različitih abnormalnosti.

Metoda segmentacije slika mozga je jedan od koraka u otkrivanju tumora mozga koji su otkriveni u MRI, PET ili SPECT slikama. Općenito, tumori se mogu svrstati u dvije kategorije: karcinomi i dobroćudni tumori. Kancerogeni tumor može biti primarni ili sekundarni. Primarni su oni koji se razvijaju unutar mozga, dok se sekundarni tumor razvija u drugim organima otkuda se dalje širi na mozak. Različiti tumori mozga imaju različite karakteristike gdje s ciljanom segmentacijom njihove granice mogu biti što preciznije određene. Tumor mozga u medicinskim slikama iz različitih izvora može se otkriti korištenjem različitih metoda segmentacije slike. Segmentacija slike je zadatak u kojem sliku treba podjeliti u više razdvojenih dijelova u kojima su elementi u jednom dijelu ili segmentu slični, ali različiti od elemenata u drugim segmentima na temelju nekih kriterija poput razine intenziteta, teksturnih karakteristika, frekvencijskih komponenti itd. Pod ciljem segmentacije u razmatranjima otkrivanja tumora mozga, smatra se jasno razlikovanje anomalija, tj. tumora, od ostatka slike.

Za segmentaciju slike predložene su različite tehnike. Neke od najčešće korištenih metoda uključuju metodu određivanja praga (eng. threshholding) metode temeljene na histogramu, metode na bazi grupiranja, metode rastućih regija, otkrivanje rubova itd. U ovom radu predložena je metoda za segmentaciju medicinske slike koja se temelji na algoritmu grupiranja. Grupiranje predstavlja tehniku strojnog učenja bez nadzora koja razdvaja podatke u odvojene grupe, gdje bi podatci unutar jedne grupe trebali biti slični po nekim mjerilima, dok su različiti od podataka u drugim grupama. Metode grupiranja pokušavaju odrediti obrasce u datom skupu podataka kako bi se još mogle izdvojiti neke dodatne informacije. Danas postoje različite metode grupiranja, uključujući hijerarhijsko grupiranje, metode grupiranja temeljene na distribuciji, DBSCAN, k-sredina, k-medijan i mnoge druge.

Jedan od najčešće korištenih algoritama grupiranja, zbog njegove jednostavnosti, je algoritam k-sredina koji predstavlja iterativni proces pretraživanja u pronalasku centroida, tj. centra grupe (Khalaf i sur., 2018 [56]). U slučaju korištenja grupiranja k–sredinom za segmentacije digitalnih slika, podaci (ili instance) koje treba podijeliti u

odvojene grupe su pikseli. Ako se algoritam k-sredina koristi kao algoritam grupiranja, tada treba definirati udaljenosti između piksela. Udaljenost se može definirati ili na temelju prostornih informacija ili na razini intenziteta piksela. Metoda segmentacije zasnovana na izrastanju područja (eng. region growing segmentation methods) često se kombinira s algoritmima grupiranja gdje se udaljenost definira kao kombinacija položaja piksela i njegove razine intenziteta kao u [3] i [62]. Elementi svake grupe određuju se njihovom udaljenošću od centroida. Svaka instanca se pridružuje onoj grupi s čijim centroidom ima najmanju udaljenost. Najbliži centroid predstavlja grupu gdje instanca pripada. Udaljenost se može definirati na brojne načine, obično kao euklidska udaljenost. Svaka iteracija sadrži dva koraka, dodjeljivanje grupe instancama i ažuriranje centroida. Novi centroidi postavljeni su na sredinu podataka unutar svake grupe.

U posljednjih nekoliko godina predstavljene su brojne metode segmentacije slika temeljenih na grupiranju. Jedan od najčešćih izbora algoritma grupiranja je algoritam k-sredina. Ozturk, Hancer i Karaboga (2015) [87] predložili su metodu segmentacije slike koja se temelji na poboljšanom algoritmu k-sredina s ABC optimizacijskim algoritmom. Predložena metoda uključivala je novu fitnes funkciju koja je koristila međuklasnu varijancu i varijancu unutar klase kao i grešku kvantizacije koja opisuje opću kvalitetu algoritma grupiranja. Predložena metoda je uspoređena s uobičajenom fitnes funkcijom i drugim najsuvremenijim metodama segmentacije. Na temelju Davies-Bouldin indeksa i XB indeksa, koji se koriste za određivanje kvalitete segmentacije, predloženi algoritam ABC nalazio je bolje grupe.

Druga metoda za segmentaciju medicinskih slika s optimizacijom roja čestica hibridizirana je s algoritmom fuzzy k-sredina i s jezgrenim algoritmom fuzzy k-sredinama predloženih od Venkatesan & Parthiban (2017) [119]. Predložena metoda testirana je na MRI slikama mozga, a kao mjera kvalitete korištena je prosječna unutar klasterska udaljenost, vrijeme računanja i Davies-Bouldin indeks. Pokazano je da su predložene hibridizirane metode imale bržu konvergenciju i manje su bile osjetljive na šumove u usporedbi s drugim postojećim metodama.

Parasar & Rathod (2017) [88] su predložili optimizaciju roja čestica (PSO) u kombinaciji s algoritmom k-sredine za segmentaciju ultrazvučne slike fetusa. Njihova metoda uspoređena je s metodom uzgoja zasijane regije optimiziranom PSO, fuzzy

k-sredinama, nadmašila ih je kada su korišteni ultrazvučni snimci sa i bez šuma.

Optimalni centroidi za fuzzy grupiranje određeni su optimizacijskim algoritmom adaptivne kemijske reakcije kod Asanambigai & Sasikala (2018) [15] . Predložena metoda segmentacije korištena je za medicinske slike, a cilj je bio otkriti abnormalna područja slike mozga, jetre, trbuha i očiju.

Liu i Qiao (2015) [73] predložili su algoritam diferencijalne evolucije hibridiziran s optimizacijom roja čestica za segmentaciju slike koja koristi algoritam fuzzy k-sredina. Predložena metoda korištena je za segmentacijske slike u HIS-ovom prostoru boja. Na temelju dobivenih rezultata metoda je sposobna za segmentaciju s minimalnim razmakom unutar klase za obje vrste slika, sa šumovima i bez.

Aljawawdeh, Imraiziq i Aljawawdeh (2017.) [8] istraživali su otkrivanje melanoma na slikama površine kože te su predložili model segmentacije i klasifikacije melanoma. Predložena metoda koristi genetski algoritam i optimizaciju rojeva čestica kako bi poboljšala segmentaciju dobivenu algoritmom k-sredina. Karakteristike za klasifikaciju izvađene su iz segmentiranih slika dok su za klasifikaciju korištene neuronske mreže. Poboljšanim algoritmom k-sredina postignuta je velika točnost klasifikacije.

Anter, Hassenian i Oliva (2019.) [11] predložili su optimizacijski algoritam pretraživanja vrana u kombinaciji s algoritmom grupiranja fuzzy k-sredina za segmentaciju slika polja kukuruza. Predložena metoda je testirana na slikama različite složenosti, perspektiva scena snimana je s različitim kamerama. Dokazano je da je optimizacija pretraživanja vrana poboljšala kvalitetu rješenja dobivenih algoritmom k-sredina i da je bila uspješna u otkrivanju redova usjeva.

5.3 Prostor pretrage za složene strukture

Najizazovnija prilagodbe algoritama inteligencije rojeva odnose se na slučajeve kada rješenja nisu brojevi, već imaju složeniju strukturu kao što su permutacije. Neke od praktičnih problema koje imaju složene strukture kao rješenja su putanje gdje je rješenje sekvencijalni niz točaka ili problem pronalaženja trokut minimalne težine, i mnogi drugi. Primjena algoritma inteligencije rojeva u ovakvim situacijama je prilično jednostavna, ali je problem na drugoj strani odnosno mora se puno više vremena potrošiti na modeliranju rješenja i funkcije cilja

5.3.1 Problem planiranja putanje

Jedan od teških optimizacijskih problema koji se često proučava u literaturi je problem planiranja putanje gdje se one mogu planirati za različite robote, letjelice, plan evakuacije u izvanrednim situacijama, i slično. Planiranje putanje predstavlja široko istraživačko područje i može se klasificirati na brojne načine. Putanja i način planiranja iste zavisi od okoline, radi li se o okolini sa statičkim ili dinamičkim preprekama; koje vrste prepreka je potrebno zaobići (prepreke koje se moraju izbjeći kao što je zid ili neka druga fizička prepreka ili su u pitanju neke regije s određenim rizikom po ispravnosti robota, letjelicu); planira li se putanja jednog robota ili više njih koji međusobno komuniciraju, da li je kretanje robota ograničeno u 2D ili 3D prostoru, itd.

Dalje u poglavlju se govori o kretanju robota ograničenog u 2D prostoru. Cilj planiranja putanje je pronaći skup točaka $S = \{A, (x_1, y_1), (x_2, y_2), \cdots, (x_d, y_d), B\}$ u globalnom prostoru pretraživanja kako bi se postigao najkraći put od početne točke A do ciljne točke B, s pretpostavkom da linija između bilo koje dvije susjedne točke ne prolazi kroz područje prepreke.

Nadalje je potrebno definirati putanju, odnosno osmisliti kako se modelira problem. Prilikom modeliranja putanje potrebno je uzeti u obzir okolinu ali i fizička ograničenja i mogućnosti robota, letjelica, ljudi, itd. S obzirom na veličinu samog robota, polumjer prepreka se širi prema veličini robota, pa se robot može smatrati točkom mase. U zavisnosti od odabranog modela putanje, različito se i prilagođavaju algoritmi inteligencije rojeva.

Putanja se može modelirati kao niz točaka pri čemu se mora paziti da taj niz predstavlja dopustivo rješenje, odnosno da putanja ne prolazi kroz zidove i slično. U tom slučaju potrebno je napraviti dodatne mehanizme. U složenom području teško je stvoriti jednu putanju koja je dopustiva. Jedna od mogućnosti je započeti s ogromnim brojem slučajnih točaka, od kojih se odabere podskup i nekako ih se poreda. Vjerojatnost da je upravo ta putanja dopustiva jednaka je nula. Prije pokretanja algoritma inteligencije rojeva, potrebno je napraviti putanju koja je dopustiva, ne bilo koju slučajnu.

Jedna od metoda pronalaženja putanje koja je dopustiva je metoda koja kombinira lokalno pretraživanja za pronalaženje najkraćeg puta u grafu s algoritmom

optimizacije inteligencije rojeva. Prvi korak u pronalaženju optimalnog puta je pronalaženje dopustivih rješenja, jer nedopustiva nisu korisna u realnim situacijama. Da bi mogućnost pronalaska optimalnog puta bila veća, koristi se ideja o stvaranju dopustivih staza slično kao što je predloženo u [30].

Put je izgrađen postepeno gdje se najprije stvaraju N slučajnih točaka (sve u prostoru pretrage i izvan prepreka). Put se zatim gradi počevši od početne točke A, a sljedeća točka staze se bira između N generiranih slučajnih točaka. Uzima se najbliža točka iz izvedivog rješenja. Nakon toga isti postupak se obavlja za odabrane točke dok se ne stigne do ciljene točke. Ovo je u osnovi Dijkstrin algoritam za pronalaženje najkraćeg puta u grafu gdje su čvorovi početna i ciljna točka i točke nasumično generirane, a rubovi su njihove veze. Rubovi koji imaju sjecište s preprekama nisu uzeti u obzir. Lokalna metoda pretraživanja je korištena za generiranje početne populacije za algoritam inteligencije rojeva. Predloženi algoritam započinje sa generiranjem N slučajnih mrežnih točaka u slobodnom prostoru (izvan prepreka). Te točke smatraju se čvorovima grafikona. Euklidska udaljenost izračunata je između svakog para čvorova i stvara se matrica udaljenosti. Kako bi se osiguralo da jedino dopustiva rješenja budu izabrana za najkraći put, udaljenost između čvorova čija bi veza dovela do nedopustivog rješenja, postavljena je na beskonačnost. Postupak lokalnog pretraživanja sličan onome predloženom u [30] koristi se za pronalaženje najkraćeg puta na tom grafu gdje su čvorovi generirane točke zajedno s početnom i ciljnom točkom.

Planiranje putanje bespilotnih letjelica je višedimenzionalni NP-teški problem. To je povezano s optimizacijom putanje leta koja je podložna višestrukim ograničenjima unutar različitih okruženja. Budući da je to NP-težak problem optimizacije, nema determinističke metode za rješavanje problema planiranja putanje u razumnom roku. Za rješavanje takvih NP-teških problema mogu se koristiti različite tehnike i metode. U posljednja dva desetljeća algoritmi nadahnuti prirodom, posebno algoritmi inteligencije rojeva, široko su se i uspješno koristili za brzo pronalaženje prihvatljivih rješenja.

Planiranje putanje važno je u mnogim zadatcima koji omogućavaju povećavanje autonomije bespilotnih letjelica (UAV). Bespilotne letjelice (UAV) ili dronovi su zrakoplovi za koje nije potreban ljudski pilot na palubi, već su daljinski pilotirani

ili samopilotirani. Mogu prevoziti razne komade opreme poput kamera, senzora, komunikacijske opreme, oružja.

U usporedbi sa standardnim avionima, oni imaju prednost u niskoj cijeni, visokoj sigurnosti, visokoj sposobnosti preživljavanja, dobrim manevarskim performansama. Budući da ne postoji bojazan da će se upotrebom UAV-a prouzročiti problem, oni se svakodnevno koriste u civilnim i modernim ratovanjima. U vojnim područjima, bespilotni borbeni zrakoplov (UCAV) ima potencijalnu, značajnu prednost nad naoružanim borcem od mogućnosti velikih manevara duboko iza neprijateljskih linija, borbenoj fleksibilnosti, radu stroja izvan mreže i tako dalje. Let UAV-a autonomno kontroliraju računala u vozilu ispod ili pilot s daljinskim upravljanjem na zemlji.

Problem planiranja putanje UAV se može opisati kao optimizacijski problem gdje je cilj pronaći optimalnu putanju od početne točke do cilja, u odnosu prema nekoj metrici. Optimalni put može se definirati na brojne načine budući da se mogu odabrati različito poželjna svojstava, na primjer, najkraća ili najglatkija putanja, minimalna potrošnja goriva, najsigurniji put itd.

Planiranje putanje, posebno za bespilotnu letjelicu (UAV) i bespilotni borbeni zrakoplov (UCAV), predstavlja aktivnu istraživačku temu od velike praktične važnosti i brojne metode su predložene za njegovo rješavanje. Posljednjih godina većina predloženih metoda temelji se na algoritmima nadahnutim prirodom. Kao što je rečeno, tradicionalne metode zbog NP-teškog problema planiranja puta UAV, nisu mogle postići razumne rezultate u razumnom vremenskom roku.

U literaturi se mogu naći brojne različite metode za planiranje putanje od kojih će neke biti spomenute. Za ovaj problem je implementirano nekoliko stohastičkih algoritama, kao što su genetski algoritam (GA) [99], optimizacija roja čestica (PSO) [68], diferencijalna evolucija (DE) [142], optimizacija na bazi biogeografije (BBO) [149], optimizacija kolonijom mrava (ACO) [38], algoritam kolonije pčela (ABC) [133], algoritam šišmiša (BA) [123], algoritam vatrometa [5] itd.

U [41] za planiranje putanje UAV-a na moru predložen je hibridni algoritam koji je kombinacija diferencijalne evolucije (DE) i kvantnog algoritma optimizacije čestica rojeva. Metoda je uspoređena sa standardnim genetskim algoritmom, diferencijalnom evolucijom, optimizacijom rojeva čestica i kvantnim PSO. Rezultati usporedbe pokazali su da je predloženi hibridni algoritam bolji u kvaliteti rješenja, robusnosti i

brzini konvergencije.

Algoritam umjetne kolonije pčela (ABC) poboljšan strategijom ravnoteže evolucije predložen je za problem planiranja putanje bespilotnih letjelica u [61]. Strategija ravnoteže evolucije uvedena je radi poboljšanja konvergencije uravnoteživanjem lokalne i globalne potrage. Ova metoda imala je bolje performanse u usporedbi sa standardnim ABC i s druga dva modificirana ABC algoritma predložena za planiranje staza u literaturi.

U [143] predložen je algoritam za optimizaciju inspiriran golubovima za rješavanje problema trodimenzionalnog planiranja putanje bespilotnog zrakoplova u dinamičnom okruženju. Model predator-plijen korišten je za poboljšanje brzine konvergencije u algoritmu nadahnutom golubom (PIO). Predložena metoda pokazala se boljom za planiranje putanje u odnosu na izvorne algoritme PSO, PIO i DE.

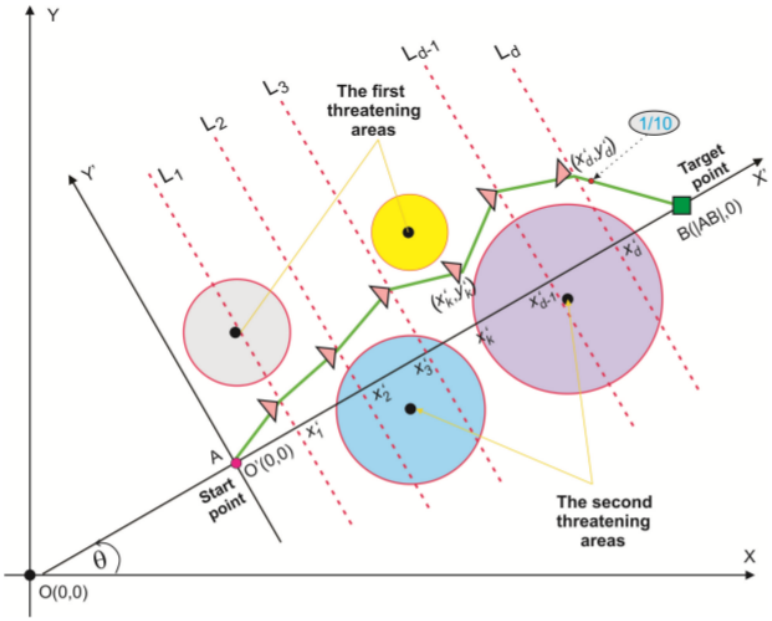
Planiranje putanje za bespilotne borbene zrakoplove riješeno je uz pomoć modificiranog algoritma krijesnica u [126]. Modifikacija je primijenjena na razmjenu informacija između krijesnica s boljim vrijednostima objektivne funkcije s čime se povećala brzina konvergencije. Predloženi algoritam je bio učinkovitiji za problem planiranja putanje u odnosu na izvorni algoritam krijesnice i za nekoliko drugih algoritama inteligenicije rojeva kao što su ACO, DE, GA, PSO i drugi.

U [124] algoritam šišmiša je izmijenjen i prilagođen za planiranje UCAV staza. Operator mutacije za stvaranje novog položaja sljepih miševa u novoj generaciji zamijenjen je u nekim slučajevima operatorom diferencijalne evolucije. Razmatrana su dva cilja za pronalaženje optimalnog puta, potrošnja goriva i izbjegavanje prijetnji. Predložena metoda je uspoređena s izvornim algoritmom šišmiša i s drugim algoritmima utemeljenim na populaciji (ACO, PSO, DE, GA, itd.). Rezultati simulacije pokazali su da je ova metoda imala bolje performanse u usporedbi s drugim spomenutim metodama.

Bolji uvid u ovaj NP-teški optimizacijski problem omogućava uvođenje matematičkog modela koji pronalazi sigurnu putanju uz pomoć koje bespilotna letjelica može slobodno letjeti s jedne točke na drugu bez da prolazi područjem prijetnje. Pretpostavka je da se zračno vozilo kreće od početne točke $A = (x_S, y_S)$ do krajnje točke $B = (x_T, y_T)$. Na putu između ovih točaka nalaze se područja koja ugrožavaju neprijatelje poput radara, topništva, raketa, itd. U ovom su modelu opasna područja

predstavljena u obliku krugova, unutar kojih će UAV biti osjetljiva na opasnost s određenom vjerojatnošću proporcionalnoj udaljenosti od centara prijetnje. Dakle, svako od opasnih područja ima određenu ocjenu rizika. Da bi bila otkrivena optimalna putanja ona mora povezivati točke $A(x_S, y_S)$ i $B(x_T, y_T)$, a istovremeno na nju ne smiju utjecati opasna područja.

Kao i obično, originalni Descartesov koordinatni sustav **XOY** pretvara se u novi rotirani **X'O'Y '**. Novi koordinatni sustav dobiva se zakretanjem starog koordinatnog sustava **XOY** u smjeru suprotnom od kazaljke na satu za kut θ, i zatim translacijom iste duž vektora $\overrightarrow{OA}$, kao što je prikazano na slici 10.



Slika 10: *Model putanje*

Kut θ označava kut između segmenta $\overline{AB}$ i pozitivnog dijela osi x, a izračunava se uz pomoć sljedeće jednadžbe:

$$\theta = \arctan\left(\frac{y_T - y_S}{x_T - x_S}\right) \tag{35}$$

Nove koordinate (x', y') su izračunate sljedećom jednadžbom:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_S \\ y_S \end{bmatrix}, \tag{36}$$

Horizontalna osa X' novog **X'O'Y'** koordinatnog sustava sadržava segment $\overline{AB}$ tako da se točke A i O' podudaraju. U novom koordinatnom sustavu **X'O'Y '** početna točka O' ima koordinate $(0,0)$, dok ciljna točka B ima koordinate $(|\overline{AB}|,0)$. Da bi se problem planiranja putanje formulirao kao d-dimenzionalni problem optimizacije, segment $\overline{AB}$ je podijeljen u $(d+1)$ jednakih podsegmenata. U svakoj od krajnjih točaka $(x'_k,0)$ generiranih podsegmenata povuću se crte $L_k(k=1,2,\cdots,d)$ koje su okomite na segment $\overline{AB}$. Nadalje, uzimanjem vrijednosti y'_k duž linija $L_k(k=1,2,\cdots,d)$, dobivena je diskretna kolekcija točaka $S=\{(x'_1,y'_1),(x'_2,y'_2),\cdots,(x'_d,y'_d)\}$. Skup S se proširuje s dvije dodatne točke $O'(0,0)$ i $B'(|\overline{AB}|,0)$. Njihovim povezivanjem dobiva se niz u kojem su te dodane točke redom na početku i na kraju. Također, kao posljedicu njihovog povezivanja u niz, dobiva se putanja leta od startne do ciljne točke. Na ovaj se način problem planiranja putanje transformira u problem optimizacije D-dimenzionalne funkcije, točnije formulira se kao problem optimizacije koordinata y'_k $(k=1,2,\cdots,d)$ kako bi se postigla podoptimalna vrijednost fitnes funkcije troška. U problemu planiranja putanje ograničenja prepreka uglavnom imaju dva aspekta: jedan je granica samoga puta; drugo je izbjegavanje prepreka. Koordinata y_i točke putanje ne može prijeći zadani gornji i donji rub putanje, također položaj ovih točaka ne može biti u području prepreke. Ako je y_i izvan granice, ponovno se nasumično odabire nova vrijednost y_i. Ako je položaj točke staze u području prepreke, mora se ponovno smjestiti u ograničenom rasponu u pozitivnom ili negativnom smjeru y_i dok se ne nađe novi položaj koji nije u području prepreke. Stoga se y-koordinata točke staze može ograničiti između gornjeg i donjeg ruba i isključiti iz područja prepreke. Osim što je potrebno osigurati da položaj točke puta ne prelazi granični raspon, treba također razmotriti jesu li prepreke na liniji između dviju susjednih točaka putanje. Ako na liniji postoji neka prepreka, putanja je nevažeća. Potrebno je ponoviti traženje nove točke putanje. Stoga se mora otkriti dostupnost puta između dvije susjedne točke staze u svakoj generaciji. Za bilo koju putanju ako je linija između y_i i y_{i+1} presijecana preprekama, potrebno je ponovno odabrati y_{i+1}. Ako se valjani y_{i+1} i dalje ne pronađe nakon određenog pokušaja, y_i bi trebao biti ponovno izabran sve dok se točka putanje ne pronađe u skladu s uvjetima ograničenja. Budući da se proces pretraživanja optimalnih vrijednosti y'_k izvodi u novom koordinatnom sustavu, koordinate središta za područja koja prijete trebaju biti određene u istom

tom koordinatnom sustavu. Za izračunavanje nove koordinate za proizvoljni centar $C(x_a, y_a)$ opasnog područja, prvo je potrebno prevesti točku C duž x -os za $-x_1$ i također za $-y_1$ duž y -osi. Zatim se rotacijom pomakne nova prevedena točka $D(x_a - x_1, y_a - y_1)$ u smjeru suprotnom od kazaljke na satu oko točke $O(0, 0)$ za kut θ. Kao rezultat, se dobije točka $C'(x'_a, y'_a)$ u novom koordinatnom sustavu **X'O'Y'**. Općenito, preslikavanje točaka iz starog u novi koordinatni sustav može se izvesti pomoću složene matrice MC koja se sastoji od matrice rotacije MR kao i matrice prijevoda MT. Dakle, $MC = MR \circ MT$, gdje se MR može izraziti pomoću Eq. 36. Dakle, $C' = MC(C)$. Za potrebe implementacije upotrijebljene su homogene koordinate kako bismo vrlo brzo i učinkovito izveli transformaciju matrice. Kao što je prikazano na Fig. 10, važno je primijetiti da se u novom koordinatnom sustavu x'_k koordinata svake točke koja pripada putanji može izračunati jednostavnom formulom $x'_k = k|AB|/(d+1)$. Stoga će putanja bespilotnih letjelica proći kroz točke zadanih u S. Sada, za fiksne vrijednost x'_k na osi X', pretpostavljamo da vrijednost ordinate na osi Y' koja se mijenja pripada zatvorenom intervalu $[a, b]$,

Da bi odredila optimalnu putanju, UAV mora izbjeći neke prijetnje i zamke. Shodno tome, potrebno je uključiti indikator sigurnosnih performansi koji se sastoji od troškova prijetnje T_C i troškova goriva F_C. Trošak prijetnje T_C definira se:

$$T_C = \frac{1}{5} \sum_{i=1}^{d+1} l_i \sum_{j=1}^{N_T} t_j \sum_{k \in K} \frac{1}{d_k^4(i,j)} \tag{37}$$

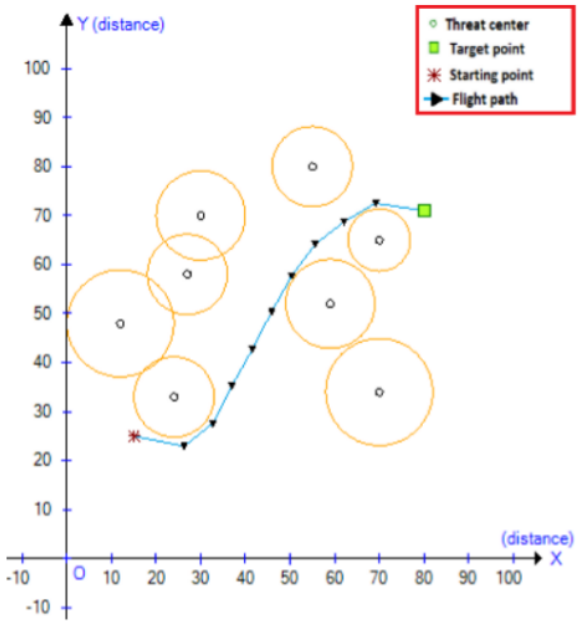
pri čemu d označava dimenziju problema, l_i je duljina i-tog podsegmenta, N_T je ukupni broj opasnih područja, t_j je stupanj prijetnje od $j-te$ prijetnje, $K = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ i $d_k(i,j)$ je udaljenost između $k-te$ točke koja pripada $i-tom$ podsegmentu i $j-tom$ centru prijetnje.

Primjer putanje dimenzije 10 je dan na slici 11.

Pod pretpostavkom da je brzina UAV konstantna, trošak goriva F_C može se formulirati kao:

$$F_C = \sum_{i=1}^{d+1} l_i \tag{38}$$

Trošak goriva ponekad se zamjenjuje duljinom staze. Sada se ukupni trošak C od UAV-a, koji putuje između početne pozicije A i željenog odredišta B, može izračunati



Slika 11: *Primjer najbolje putanje leta kada je dimenzija problema jednaka 10 (d=10)*

iz ponderirane sume prijetnji i troškova goriva na sljedeći način:

$$C = \lambda \cdot T_C + (1 - \lambda) \cdot F_C \tag{39}$$

pri čemu je $\lambda \in [0, 1]$ varijabla koja se koristi za uspostavljanje ravnoteže između stupnja izloženosti prijetnji i potrošnje goriva. Za usporedbu algoritama, parametar λ bit će postavljen na 0,5. Konačno, optimalno planiranje putanje UAV-a može se postići minimiziranjem Eq. 39.

5.3.2 Trinagulacija minimalne težine

U računalnoj geometriji postoje mnogi izazovi koji su teški problemi optimizacije i za njih nisu poznati polinomni algoritmi. Jedan od takvih izazova je problem pronalaska triangulacije minimalne težine. Temelji se na traženju minimalnog zbroja duljina rubova preko svih mogućih trokuta za određeni skup s n točaka u dvodimenzionalnom prostoru. Stoga, potrebno vrijeme računanja za algoritam iscrpnog pretraživanja raste eksponencijalno s brojem točaka u ravnini. Budući da je riječ o veoma zahtjevnom problemu, potrebno vrijeme računanja za algoritam iscrpnoga pretraživanja raste eksponencijalno s brojem točaka u dvodimenzionalnom

prostoru.

Problem pronalaženja triangulacije minimalne težine datira još iz 1970-ih i nazvan je potencijalno najdugovječnijim otvorenim problemom u računalnoj geometriji. Uveli su ga Garey i Johnson na svom popisu otvorenih problema [43]. Skoro 30 godina definiranja problema, Mulzer i Rote su 2008. dokazali da je to NP-težak problem [80]. U računalnoj geometriji, problemi optimizacije vezani za određene geometrijske konfiguracije zanimljivi su za istraživanje zbog njihove upotrebe u mnogim poljima primjene. Jedan od takvih problema je planarna triangulacija koja je postala vrlo popularna prvenstveno zbog velikog broja aplikacija poput metoda konačnih elemenata, računalno potpomognutog geometrijskog dizajna, interpolacije površine, računanja u numeričkoj analizi, računalne grafike, robotike, računalnog vida i sintezi slike, vidljivosti, detekcija kinetičkih sudara, rigidnost, i mnogih drugih. Također, drugi značajni računalni problemi geometrije, poput geometrijskog pretraživanja i presijecanja poliedra, koriste ravninske triangulacije kao fazu predobrade.

Rješavanje problema triangulacije mnogokuta dovelo je do otkrića Catalanovih brojeva. Zapravo, pokušavajući pronaći broj različitih načina da se triangulira konveksni mnogokut, Euler je prvi otkrio da taj broj, svaki put, uzima određenu vrijednost u odnosu na neke vrhove. Iako ga je Euler prvi otkrio, taj je broj kasnije postao poznat kao Catalanov broj, u čast belgijskog matematičara. Veza između Catalanovih brojeva C_n i broja različitih načina trianguliranja konveksnog mnogokuta od n vrhova dana je:

$$T_n = C + n - 2, \quad n \geq 3, \tag{40}$$

pri čemu T_n stoji za broj mogućih trokuta konveksnog mnogokuta, a C_n je definiran kao:

$$C_n = \frac{(2n)!}{(n+1)!\ n!}. \tag{41}$$

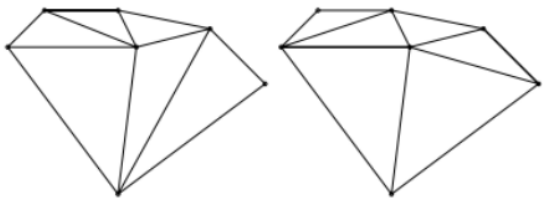
Uzimajući u obzir brojnik u jednakosti (41) može se vidjeti da je pronalaženje svih mogućih trokuta određenog mnogokuta ili, čak štoviše, određenog skupa točaka, eksponencijalni kombinatorički problem i NP- težak problem, kao i pronalaženje onog s najmanjom sumom težina rubova [80].

Napomene i uobičajeni pojmovi koji se odnose na triangulaciju su sljedeći. Skupovi vrhova i rubova grafa G označeni su s V(G) i E(G). Analogno, skupovi vrhova i rubova triangulacije T označeni su s V(T) i E(T). H(S) je oznaka za trup skupa točaka S. Problem triangulacije skupa točaka S svodi se na pronalaženje ravninskog grafa G gdje je V(G) = S i svako lice G, osim vanjskog, ograničeno je s točno tri ruba. Upućivanje na triangulaciju T ekvivalentno je određenom ravninskom grafu s gore navedenim atributima. Težina triangulacije T, označena kao težina wt(T) i predstavlja zbroj duljina rubova triangulacije. Problem pronalaska trokuta minimalne težine svodi se na pronalaženje trokuta s minimalnim zbrojem duljina rubova. Euklidska udaljenost za izračunavanje duljina ruba najprirodniji je izbor, ali često se, uključujući i u ovom radu, za jednostavnije proračune koristi kvadrat euklidske udaljenosti.

Da bi se riješio ovaj problem, neki algoritmi, kao što su linearno programiranje [141], graf najbližeg susjeda [58], pohlepno pretraživanje s lokalnom pretragom [36] i drugi su bili korišteni ranije.

Primjena algoritama inspiranih prirodom na ovaj problem je vrlo kompliciran. Za razliku od prethodno opisanog problema, traženja putanje gdje je samo rješenje specijalna struktura, u ovom problemu postoje brojna pitanja i problemi koji se moraju prevazići kako bi se uspješno primenili algoritmi inspirirani prirodom, posebno algoritmi inteligencije rojeva.

Prje svega potrebno je definirati šta je jedno rješenje? Jedno rješenje mora sadržati informaciju o trokutima koji su deo trijangulacije minimalne težine. Najčešća reprezentacija rješenja triangulacije je graf gde su točke u prostoru koje potrebno je triangulisati čvorovi grafa te njih nije potrebno pamtiti unutar rješenja. Ono što je bitno za rješenje jeste spisak rubova grafa. Na slici 12 prikazane su dve različite trijangulacije istog skupa podataka.



Slika 12: *Primjeri različitih triangulacija istig skupa točaka*

Obzirom da su rješenja spisak rubova, odnosno spisak parova čvorova grafa koji čine rubove trokuta koji su deo rješenja triangulacije, kako bi triangulacija bila uspešna odnosno da kako bi rješnje bilo dopustivo potrebno je da su svi čvorovi grafa teme točno jednog ruba i da dati skup točaka biva razdijeljen na trokute spajanjem tih rubova. Iz ovoga sledi da drugi problem koji bi trebalo riješiti je problem pronalaženja dopustivih rješenja.

Kada je definirana struktura rješenja i način generiranja slučajnih dopustivih rješenja, tada je potrebno definirati i operatore koji generiraju bliska rješenja odnosno operator za potrebe ekspoatacije. Obzirom na strukturu rjepenja standardne operacije algoritama inteligencije ne mogu se koristiti. Za problem triangulacije minimalne težine postoje različiti operatori eklpoaracije u literaturi. Jedan od standardnih metoda je takozvana metoda okretanje rubova (engl. edge flipping). Prvo se rubovi okruženi konveksnim pravokutnicima od svih unutarnjih rubova u rješenju filtriraju. Među novo filtriranim rubovima, samo oni s većom težinom od njihovih dijagonala suprotnih dijelova u okolnim pravokutnicima klasificiraju se kao oni koji se mogu popraviti. Nakon toga, među popravljivim rubovima jedan se bira nasumično i zamjenjuje ga se s njegovom dijagonalom suprotne strane. Ovaj proces generira rješenje manje težine nakon svake izvedbe. Obzirom da se od svih rubova koji mogu smanjiti težinu triangualcije bira jedna nasumično tada ovo predstavlja fazu u kojoj se primjenjuje stohastičko penjanje uzbrdo.

Ukoliko se pogleda šta je izmjenjeno u algoritmu, može se zaključiti da jedini deo originalnog algoritma je upravo inteligencija roja to jest kako rješenja međusobno utiču jedna na druge i koja rješenja se kombiniraju, kada se vrši eksploracija kada eksploatacije i slično.

U literaturi je predloženo nekoliko pristupa problemu triangulacije minimalne težine. U brojnim studijama korišteni su modificirani i prilagođeni genetski algoritmi. U [128] predložen je kvantni genetski algoritam za triangulaciju minimalne težine, a dokazano je da je bolji od pohlepne metode. Genetski algoritam u kombinaciji s odabirom kaljenja predložen je u [71]. Odabir kaljenja sprječavao je ranu konvergenciju, čime je učinkovitost predložene metode bila bolja u odnosu na genetski algoritam.

Osim genetskih algoritama, korišteni su i algoritmi inteligencije rojeva. U [37] algoritam optimizacijom kolonijom mrava primijenjen je na triangulaciju najmanje

težine kao i na pseudo - triangulaciju minimalne težine. Optimizacija kolonijom mrava nadmašila je pohlepne i algoritme simuliranog kaljenja.

U [58] opisuje se odnos grafa najbližeg susjeda (NNG) prema lokalno minimalnoj triangulaciji (LMT) i pokazuje da, iako se može dokazati da NNG nije podgrupa LMT-a, u većini slučajeva LMT sadrži sve ili gotovo sve rubove NNG-a.

U [36] razmatran je problem pseudo-triangulacije minimalne težine (MWPT) danog skupa od n točaka u ravnini. Ovaj rad prikazuje kako se metaheuristička metoda optimizacije kolonijom mrava (ACO) može upotrijebiti za pronalaženje optimalnih pseudo-triangulacija minimalne težine. Za eksperimentalnu studiju stvoren je skup primjeraka za MWPT jer u literaturi nisu pronađene referentne vrijednosti za ove probleme.

6 Rezultati istraživanja prilagođavanja algoritama inteligencije rojeva za praktične probleme sa različitim prostorima pretrage

U ovom poglavlju je dan pregled eksperimentalnih rezultata dobivenih primjenom različitih algoritama inteligencije rojeva na probleme opisane u prethodnom poglavlju. Postignuti rezultati su objavljeni u časopisima sa SCI liste ili na priznatim međunarodnim konferencijama i to u slijedećim radovima:

1. Tuba, E., Ribic, I., Capor Hrosik, R., & Tuba, M. (2017). Support Vector Machine Optimized by Elephant Herding Algorithm for Erythemato–Squamous Diseases Detection, *Procedia Computer Science*, 122, Elsevier, (pp. 916-923) (prezentirano na 5th Information Technology and Quantitative Management (ITQM 2017), New Delhi, India)
2. Tuba, E., Jovanovic, R., Capor Hrosik, R., Alihodzic A.,& Tuba, M. (2018). Web Intelligence Data Clustering by Bare Bone Fireworks Algorithm Combined with K-Means, 8th ACM International Conference on Web Intelligence, Mining and Semantics, (pp. 1–8).
3. Capor Hrosik, R., Tuba, E., Dolicanin, E., Jovanovic, R., & Tuba, M. (2019). Brain image segmentation Based on Firefly Algorithm Combined with K-means Clustering. *Studies in Informatics and Control*, 28(2), (pp. 167-176)
4. Alihodzic A., Tuba, E., Capor Hrosik, R., Dolicanin, E. & Tuba, M. (2017). Unmanned Aerial Vehicle Path Planning problem by adjusted elephant herding optimization , IEEE 25th Telecommunication Forum (TELFOR), (pp.1–4).
5. Alihodzic, A., Smajlovic, H., Tuba, E., Capor Hrosik, R., & Tuba, M. (2018). Adjusted Artificial Bee Colony Algorithm for the Minimum Weight Triangulation, *Advances in Intelligent Systems and Computing: Harmony Search and Nature Inspired Optimization Algorithms*, 741, Springer, (pp. 305-317). (prezentirano na 4th International Conference on Harmony search, Soft Computing and Applications, Guargon, India)

6.1 Rezultati prilagodjavanja algoritama za prostor realnih rješenja

6.1.1 Optimizacija stroja potpornih vektora za otkrivanje eritematoloških–skvamoznih bolesti algoritmom krda slonova

Algoritmi strojnog učenja koriste se u brojnim područjima, a medicina je jedan od njih. Jedna od primjena stroj potpornih vektora, metode strojnog učenja, na praktične probleme je otkrivanje eritematoloških–skvamoznih bolesti. Automatska dijagnoza ili otkrivanje različitih bolesti na temelju popisa simptoma može drastično poboljšati i ubrzati proces dijagnostike. Određivanje dijagnoze u ranijim fazama dava bolje rezultate u procesu ozdravljenja. To je i jedan od osnovnih razlog ulaska računalnih znanosti u područje medicine.

Za primjenu SVM na praktične probleme predložena je metoda za automatsku klasifikaciju eritematoloških–skvamoznih bolesti u dermatologiji. Eritematološke–skvamozne bolesti su skupina različitih dermatoloških poremećaja poput psorijaze, seboreičnog dermatitisa, lichen planusa, pityriasis rosea, kroničnog dermatitisa i pityriasis rubra pilaris. Razlike između ovih bolesti nisu očite i vrlo ih je teško razlikovati čak i biopsijom. Uz to, svaka bolest ima različite simptome u ranoj i kasnijoj fazi. Tih šest eritematoloških–skvamoznih bolesti, koje je vrlo teško razlikovati, klasificirano je pomoću algoritma potpornih vektora (eng. support vector machine, SVM). Stroj potpornih vektora je predložena za njihovo točno određivanje. Da bi se postigli najbolji mogući rezultati potrebno je prilagoditi dva parametra stroja potpornih vektora. Ovaj problem predstavlja težak optimizacijski problem, jer parametri mogu uzeti bilo koju realnu vrijednost, a ciljna funkcija ima brojne lokalne optimume. Brojne uspješne primene optimizirane stroja potpornih vektora mogu se pronaći u literaturi [104], [109].

Otkrivanje eritematoloških–skvamoznih bolesti je aktivno istraživačko polje. U literaturi se mogu naći brojne različite metode za njihovo otkrivanje od kojih će neke biti spomenute. Dvije glavne skupine se mogu prepoznati u ovim studijama. Prva skupina predlaže različite klasifikatore, njihova poboljšanja i kombinacije, dok druga skupina se koncentrira na odabir skupa najboljih obilježja koje razlikuju bolesti.

U [14] predložena je selekcija obilježja na temelju dobivenih informacija i uzastop-

nih primena metode gde se broj odabranih atributa menja (engl. floating search). Metode filtera i omotača su kombinirane kako bi se odabrao optimalni skup obilježja, dok je za klasifikaciju korišten naivni Bayesov klasifikator. Ova hibridna i pohlepna metoda odabira obilježja pokazala je obećavajuće rezultate.

Druga metoda odabira obilježja predložena je u [132] gdje se za klasifikaciju koristio SVM. Predložena metoda slična je prethodno opisanoj, ali umjesto kombiniranja dobivanja informacija s pohlepnim algoritmom SBFS, kombinira SBFS i F-score. F-score se najprije poboljšao, a zatim koristio za ocjenu kriterija metode filtriranja. SBFS je sustav ocjenjivanja za metode omota. Optimalni skup obilježja tada je dan SVM za ulazne vektore.

U [1] optimiziranom klasifikatoru, zajedno s tehnikom odabira obilježja, predloženo je automatsko otkrivanje eritematoloških - skvamoznih bolesti. Odabir obilježja izvršen je pravilima udruživanja. Kad je iz originalnog skupa obilježja odabran optimalan podskup svojstava, algoritam optimizacije rojeva čestica korišten je za pronalaženje optimalnih parametara za SVM. Optimizirani SVM tada je korišten za razvrstavanje eritematoloških - skvamoznih bolesti.

Odabir obilježja prema genetskom algoritmu i klasifikaciji po Bayesovoj mreži za dijagnozu eritematoloških - skvamoznih bolesti predstavljen je u [86]. Tijekom faze treninga korišten je genetski algoritam za traženje najboljeg skupa obilježja i za poboljšanje točnosti klasifikacije dobivene Bayesovom mrežom. Metoda je uspoređena s drugim pristupima iz literature i pokazala je obećavajuće rezultate.

Usporedba dva dobro poznata algoritma strojnog učenja, stroja za ekstremno učenje (ELM) i umjetne neuronske mreže (ANN) za otkrivanje eritematoloških - skvamoznih bolesti prikazana je u [85]. Obje su metode testirane na standardnoj bazi podataka. ELM je nadmašio ANN i pokazao dobru generalizacijsku sposobnost s velikom brzinom učenja.

U [117] predložena je metoda grupiranja za otkrivanje eritematoloških - skvamoznih bolesti. Pet različitih bolesti diferencirano je korištenjem algoritma grupiranja k-sredina.

U [91] predložena je metoda za automatsko otkrivanje eritematoloških - skvamoznih bolesti na temelju fuzzy extreme learning machine. Da bi se dobila što točnija klasifikacija, podatci su prethodno obrađeni. Nakon toga dobivene su neizrazite vri-

jednosti i korištene su kao ulaz za stroj za ekstremno učenje. Kombinacija neizrazite logike i stroja za ekstremno učenje omogućila je točnije rezultate s manje računalne složenosti u usporedbi s drugim metodama u literaturi.

Druga zanimljiva metoda za automatsko otkrivanje eritemato-skvamoznih bolesti predložena je u [18]. Predloženo je korisničko sučelje gdje je odabir obilježja izvršen algoritmom Apriori, zatim su za klasificiranje bolesti korišteni AdaBoost i hibridni AdaBoost SVM klasifikator. Objavljeno je da predložena metoda poboljšava vrijeme računanja u usporedbi s drugim metodama u literaturi.

U ovoj disertaciji analiziran je kvalitet jedanog od novijih algoritama inteligencije rojeva, algoritam optimizacije krda slonova (engl. elephant herding optimization, EHO). Ovaj algorita korišten je za pronalaženje optimalnih metaparametara stroja potpornih vektora odnosno za pronalaženje najboljeg para vrijednosti za parametre C i γ. Funkcija cilja je točnost kreiranog modela. Uz pomoć optimiziranog SVM-a dalje se utvrđivala točnost dijagnoze tj. o kojoj je eritematološkoj–skvamoznoj bolesti riječ. Točnost predložene metode uspoređena je s drugim pristupima iz literature koristeći standardni skup podataka. U svim eksperimentima postignuti su bolji rezultati.

Predložena metoda za otkrivanje eritematološko–skvamozne bolesti pomoću stojeva potpornih vektora optimizirana algoritmom krda slonova implementirana je u Matlab R2016a. Simulacije za ovu metodu izvedene su na platformi s Intel ® Core ™ i7-3770K CPU na 4GHz, 8GB RAM-a, operativni sistem Windows 10 Professional.

Kako bi se utvrdila kvaliteta predložene metode izvršena su testiranja na standardnom skupu podataka ErythematoSquamouse bolesti s kalifornijskog sveučilišta Irvine (University of California, Irvine, UCI) respozitorija za podataka za strojno učenje [69]. Ovaj skup podataka sadrži vektore atributa za 366 pacijenata. Za svakog pacijenta zabilježena su 34 atributa, 12 kliničkih i 22 histopatološka. Od 34 atributa, 33 vrednuje se linearne a jedan od njih je nominalna. Diferenciranje dijagnoza eritemato–skvamoznih bolesti je realan problem u dermatologiji. Sve bolesti se razlikuju po kliničkim atributima eriteme i skaliranja s vrlo malim razlikama. Bolesti koje spadaju u ovu skupinu su psorijaza, seboreični dermatitis, lichen planus, pityriasis rosea, kronični dermatitis i pityriasis rubra pilaris. Za dijagnozu je obično potrebna biopsija, ali nažalost ove bolesti dijele mnoge histopatološke atribute. Još

jedna poteškoća za razlikovanje ovih bolesti je da bolest može u početku pokazati obilježja druge bolesti a tek u slijedećem stadiju prikazati neke očekivane simptome. Pacijenti su prvi put klinički ocijenjeni s 12 karakteristika. Nakon toga uzeti su uzorci kože za procjenu 22 histopatoloških atributa. Vrijednosti histopatoloških obilježja određuju se analizom uzoraka pod mikroskopom. Klinička svojstva prikupljena za svakog pacijenta (instancu unutar baze) su eritem, skaliranje, granice, svrbež, fenomen koebnera, poligonalne papule, folikularne papule, zahvaćanje oralne sluznice, zahvaćanje koljena i lakta, zahvaćenost vlasišta, obiteljska povijest, dob. Skup podataka sadrži nekoliko nedostajućih vrijednosti za dob pacijenata. Nedostajuće vrijednosti zamijenjene su srednjom vrijednošću dobi drugih pacijenata za koje je taj podatak poznat. Histopatološka obilježja su inkontinencija melanina, eozinofili u infiltratu, PNL infiltrata, fibroza papilarnog dermisa, egzocitoza, akanthoza, hiperkeratoza, parakeratoza, klađenje rete grebena, izduženje rete grebena, stanjivanje suprapapilarne epiderme, spongiformna pustula, mikroprocesor munro, žarišna hipergranuloza, nestanak zrnatog sloja, vakuolizacija i oštećenje bazalnog sloja, spongioza, izgled pile-zuba ostataka, čep od folikularnog roga, perifolikularna parakeratoza, upalni monoluklearni inflitrat i infiltrata poput trake. Sva histološka obilježja imaju vrijednost 0, 1, 2 ili 3. Distribucija instanci po klasama unutar korištenog skupa podataka prikazana je u Tablici 2.

Tabela 2: *Distribucija instanci u skupu podataka*

Broj klase	**Klasa**	**Broj instanci**
1	Psorijaza	112
2	Seboreični dermatitis	61
3	Lichen planus	72
4	Pityriasis rosea	49
5	Kronični dermatitis	52
6	Pityriasis rubra pilaris	20

Parametri za algoritam krda slonova postavljeni su empirijski. Inicijalna testiranja uključuju preporučene vrijednosti parametara iz originalnog rada gdje je algoritam predstavljen. Nakon testiranja s različitim parametrima zaključeno je da su slijedeće vrijednosti najpogodnije za podešavanje metaparametara stroja potpornih vektora za klasifikaciju kožnih bolesti na osnovu kliničkih i histopatoloških obilježja. Broj slonova odnosno veličina populacije je postavljena na 50 i populacija se u svakoj iteraciji dijeli u 5 klanova te broj rješenja u svakom klasteru je 10. Kriterij za

zaustavljanje u ovom istraživanju je broj evaluacija funkcije cilja. Maksimalan broj evaluacije funkcije cilja je postavljen na 10 000. Parametar α koji određuje utjecaj najboljeg rješenja populacije postavljen je na 0.7. Parametar koji kontrolira utjecaj centra klana što je najbolje rješenje unutar jednog klastera je $\beta = 0.1$. Funkcija cilja u ovom istraživanju je točnost klasifikacije. Točnost klasifikacije izračunata je kao postotak ispravno klasificiranih instanci. Prostor za pretraživanje parametara C i γ postavljen je za eksponencijalno rastuće sekvence kako je predloženo u [48]. To se pokazalo kao praktična metoda za prepoznavanje dobrih vrijednosti parametara. Generirane vrijednosti od strane EHO algoritma korištene su za $\log_2$ - prostor. Za parametar C vrijednosti su tražene u intervalu $\log_2 C \in [-5, 20]$, a za parametar γ interval pretrage bio je $\log_2 \gamma \in [-15, 10]$.

Provjera kvalitete predložene metode učinjena je usporedbom rezultata postignutih s drugim metodama iz literature. Predloženi algoritam uspoređen je s metodom predloženom u [85] gdje su za klasifikaciju eritematoloških–skvamoznih kožnih bolesti korišteni stroj za ekstremno učenje (engl. extreme learning machine, ELM) i umjetna neuronska mreža (engl. artificial neural netework, ANN). Eksperimenti u ovom istraživanju su organizirani na isti način kao i u [85] čime je osigurana pravedna usporedba postignutih rezultata. Korišten skup podataka je bio podijeljen na trening i test podatke. Predložena metoda je kao i u radu [85] testirana na četiri različita uzoraka. Uvedena su četiri različita omjera podataka u skupu za trening i testiranje: 80%:20%, 70%:30%, 60%:40% i 50%:50%. Skup podataka podijeljen je korištenjem principa stratificiranog uzorkovanja gdje je isti postotak unutar svake klase izdvojen u trening i test skup.

Uspoređeni rezultati predložene metode i metoda iz literature su prikazani u Tablici 3. Tablica 3 prikazuje točnosti klasifikacije odnosno točnost razlikovanja bolesti dobivene predloženom metodom i metodama prikazanih u [85]. Bolji rezultati su podebljani.

U [85] već je objavljeno da stroj za ekstremno učenje pokazuje bolje rezultate u usporedbi s umjetnom neurološkom mrežom. ELM je nadmašio ANN u svim slučajevima, kako u procesu treniranja na trening skupu tako i prilikom testiranja na test skupu. U metodi predloženoj u ovom istraživanju, stroj potpornih vektora optimiziran algoritmom krda slonova (EHO-SVM), točnost u procesu treniranja

Tabela 3: *Rezultati usporedbe za različite omjere trening i test skupa*

% instanci	Skup	ANN	ELM	EHO-SVM
80	Trening	86.86	**100.00**	**100.00**
20	Test	82.74	94.50	**99.07**
70	Trening	90.08	**100.00**	**100.00**
30	Test	79.17	98.17	**98.91**
60	Trening	89.32	**100.00**	**100.00**
40	Test	77.26	96.58	**98.63**
50	Trening	89.56	**100.00**	**100.00**
50	Test	78.09	98.36	**98.61**

korištenjem skupa treninga bila je u svim slučajevima 100%. To je također bio slučaj i s ELM metodom predloženom u [85]. Ovo samo po sebi ne bi bio rezultat pošto bi se moglo protumačiti kao pretreniran model (engl. overfitting) te je potrebno analizirati i rezultate postignute na test skupu. Predložena metoda u ovom istraživanju nadmašila je točnost dobivenu na testnom skupu u svim slučajevima.

Istaknuto je u [85] da se točnost ELM metode dobivene na testnom skupu povećava kada se veličina skupa za trening smanji. U slučaju metode predložene u ovom istraživanju, EHO-SVM, situacija se preokreće, odnosno točnost na testnom skupu povećava se s brojem slučajeva u skupu za trening. Najveća preciznost postignuta je odnosom trening:test = 80:20% dok je drugi najbolji bio omjer 70:30%. Postignute točnosti na skupu ispitivanja bile su redom 99.07% i 98.91%. To znači da predložena metoda ne prevladava model. S više instanci u setu za treniranje stroj potpornih vektora može prepoznati više sličnosti među instancama iste klase i više razlika između instanci iz različitih klasa. S druge strane, kada se koristi manje podataka za treniranje pronalazi se lošiji model i nisu prepoznate sve karakteristike unutar klasa, a klasifikacija nepoznatih podataka je lošija nego u prethodnom slučaju. Budući da je točnost dobivena tijekom procesa treniranja 100%, to znači da je EHO-SVM na temelju datih podataka pronašao dobar model. Najveća razlika između točnosti postignutih na testnom skupu je u slučaju podjele trening:test skupa u omjeru 80:20%. U ovom slučaju, metoda predložena u ovoj disertaciji je postigla 99.07% dok je ELM metoda postigla samo 94.50%. S druge strane, najsličniji rezultati na testnom skupu su postignuti u slučaju podijele skupa u omjeru 50/50% kada je predložena EHO-SVM metoda postigla točnost 98.61% a ELM metoda 98.36%.

Najbolji rezultat postignut od strane ANN metoda bio je 90.08% prilikom procesa

treniranja (odnosno na trening skupu) kada je 70% podataka korišteno kada je postignuta točnost od 90.08% dok je na test skupu najveća točnost od 82.74% postignuta prilikom korišćenja 20% podataka. ANN metoda prikazana u [85] u svim slučajevima postigla je točnost manju od 80% osim u slučaju podijele 80/20% kada je, kao što je pomenuto, točnost na testnom skupu bila 82.74% što je opet daleko lošije od druge dvije metode, ELM i predložene EHO-SVM. ELM metoda je postigla 100% točnost u fazi treninga u svim slučajevima, dok je najbolja točnost iznosila 98.36% s testnim setom koji sadrži 50% podataka iz cijelog skupa podataka. Predložena SVM-EHO metoda je također postigla maksimalnu točnost u fazi treniranja, a najbolja točnost na testnom skupu od 99.07% bila je dobivena s 20% podataka iz cijelog skupa podataka. Na osnovu prikazanih rezultata postignutih predloženom EHO-SVM metodom i njihovom usporedbom sa rezultatima iz literature može se zaključiti da je predložena metoda preciznija u usporedbi s ANN-om i ELM-om. Nadalje analizom rezultata postignutih EHO-SVM metodom može se zaključiti da predložena metoda nije podložna pretreniranju te se može reći da predstavlja stvaran siguran model za klasificiranje nepoznatih instanci odnosno za klasificiranje kožnih bolesti na osnovu 33 atributa.

6.1.2 Grupiranje podataka web inteligencije ogoljenim algoritmom vatrometa u kombinaciji s K-sredinama

Jedna od primjena algoritma k-sredina na praktične probleme je primjena na web inteligenciju. Rudarenje podataka (eng. data mining) i grupiranje važni su elementi raznovrsnih primjena u različitim područjima. Jedno od područja u kojima se grupiranje često koristi je web inteligencija, koja danas predstavlja važno istraživačko područje. Podatci prikupljeni s interneta obično su vrlo složeni, dinamični, bez strukture i prilično veliki. Tradicionalne tehnike grupiranja nisu dovoljno učinkovite i potrebno ih je unaprijediti.

Web inteligencija predstavlja relativno novo područje znanstvenog istraživanja. Izraz web intelligence osnovan je 2000. godine [148] i danas je jedno od perspektivnijih područja u informacijskim tehnologijama i računarskim znanostima. Web inteligencija ima specifičan fokus na podatke i aplikacije s web platformi. Uobičajeni zadatci u ovom polju su klasifikacija teksta, grupiranje web dokumenata, preporuke za

e-trgovinu, itd.

Web inteligencija uključuje brojna područja kao što su rudarenje podataka, prepoznavanje uzoraka, strojno učenje, analize predviđanja, itd. U posljednja dva desetljeća različiti zadatci otkrivanja znanja na temelju podataka prikupljenih s weba, uobičajeno nazvanih podatcima web inteligencije, bili su predmet mnogih znanstvenih istraživanja [77], [90]. Podatci web inteligencije postaju sve složeniji, obično su vrlo nestrukturirani sa složenim atributima, zbog čega ih je teško analizirati, dok je s druge strane potrebna učinkovita analiza web elemenata [76]. Da bi se mogli nositi s takvim podatcima, potrebno je poboljšano strojno učenje, rudarenje podataka, prepoznavanje uzoraka i drugi algoritmi.

Mnoge aplikacije za web inteligenciju temelje se na algoritmima rudarenja podataka, čime se nedavno počeo koristiti i izraz web rudarenja (eng. web mining). Čest zadatak u web rudarenju je grupiranje. Standardni algoritmi grupiranja kao što su k-sredina, k-najbliže susjedstvo, DBSCAN, hijerarhijsko grupiranje i drugi, obično nisu dovoljno dobri za podatke web inteligencije, što je rezultiralo brojnim poboljšanjima ovih algoritama. Metode grupiranja mogu se poboljšati uvođenjem mogućnosti optimizacije.

U literaturi se mogu naći brojne različite metode za grupiranje podataka web inteligencije od kojih su neke spomenute u nastavku. Metoda za grupiranje podataka putem pretraživanja weba predložena je u [28]. Ta metoda temeljila se na nekoliko algoritama za rudaranje podataka, strojno učenje i optimizaciju. Modificirani algoritam pretraživanja kukavice (eng. cuckoo search) korišten je za pronalaženje optimalnih grupa. Umjesto korištenja leta Levi, koji je izvorno bio dio algoritma pretraživanja kukavice, za generiranje novih rješenja na temelju prethodnih koristila se metoda razdvajanja i spajanja. Rješenja su evaluirana algoritmom k-sredina, gdje se mjeru sličnosti uzima Bayesov informacijski kriterij. Predložena metoda omogućava i dinamičko prilagođavanje broja grupa.

Metoda za učinkovito preporučivanje web stranica predložena je u [40]. Predložena metoda temelji se na algoritmu harmonijskog pretraživanja i algoritmu grupiranja k–sredina te je korištena za podjelu web podataka binarnih sesija na unaprijed definirani broj grupa. Algoritam harmonijskog pretraživanja prilagođen je za rješavanje problema kombinatorne optimizacije, tj. kako bi se utvrdilo koji

podatci pripadaju kojoj grupi. Izlaz algoritma harmonijskog pretraživanja dava skoro optimalno rješenje, a algoritam k–sredina korišten je za poboljšanje kvalitete pronađenog rješenja. Predložena metoda je testirana na stvarnim skupovima podataka, a rezultati su uspoređeni s drugim pristupima iz literature. Dobiveni su bolji rezultati u pogledu F-mjere i VC za preporuku web stranica.

U [42] predložena je metoda grupiranja web dokumenata. Predložena metoda razmatrala je oboje, sadržaj weba i veze na web stranicama. Hibridni pristup kombinira konceptualni model rudarenja s hijerarhijsko aglomeracijskim grupiranjem. Razmotrivši poveznice na web stranicama, kvaliteta grupa povećala se u usporedbi s drugim pristupima iz literature.

Bisekcijski k-sredina algoritam za grupiranje web korisnika predložen je u [89]. Predložena metoda je bila kombinacija algoritma k-sredina i hijerarhijskog grupiranja. Algoritam je iterativnog procesa koji započinje jednom grupom koja je podijeljena u dvije grupe algoritmom k-sredina u svakoj iteraciji. Predložena metoda osigurala je manje klastere u usporedbi s originalnim algoritmom k–sredina, što je rezultiralo grupama prilično različitih veličina. Također prikazana je usporedba u točnosti i vremenu izvođenja. Raščlanjavanje algoritma k–sredina nadmašio je originalni algoritam k–sredina za sve primjere ispitivanja.

U [81] predložena je metoda koja se temelji na pravilu pridruživanja i algoritma k-sredina za grupiranje XML dokumenata. Prvi korak metode bio je pronaći obilježja XML dokumenata. Umjesto predstavljanja svakog dokumenta zasebno, predložen je pronalazak svojstava za skupove sličnih dokumenata po pravilu pridruživanja. Nakon ekstrakcije obilježja, grupiranje je izvedeno algoritmom k–sredina. Predložena metoda pokazala se prikladnom za pronalaženje grupa za XML podatke.

Metoda za preporučivanje novinskih članaka utemeljena na obilježjima korisnika i teksta, predložena je u [20]. Algoritam k-sredina korišten je za opisivanje podataka grupe s nekoliko tekstualnih i korisničkih obilježja. Algoritam grupiranja kombiniran je s drugim tehnikama pretraživanja informacija poput kategorizacije teksta i rezimiranja. Predložena metoda nadmašila je metode u kojima tehnike pretraživanja podataka nisu kombinirane.

Metoda rudarenja podataka o webu i metoda za preporučivanje koja se temelji na k-najbližim susjedima predložena je u [2]. Relevantne informacije prikupljene

su sa stvarno jednostavnih web čitača (RSS), koji hvataju korisnike koji kliknu na stream podatke. Prikupljeni podatci oblikovani su i grupirani kako bi se koristili kao ulaz za metodu klasifikacije k-najbližih susjeda. Algoritam je prilagođen kako bi se koristio i u vanmrežnom i u realnom vremenu. Pokazalo se da predložena metoda je dobra za slučajeve kad je prethodno bilo malo znanja ili uopće nije bilo znanja o raspodjeli podataka.

U [9] predložena je metoda za predviđanje vrste korisnika i uzorka za navigaciju. Tri su tipa korisnika prepoznata po metodi koja uključuje predobradu podataka, klasifikaciju i otkrivanje uzoraka. Predobrada uključivala je uklanjanje lokalnog i globalnog šuma, čišćenje podataka iz neuspjelih zapisnika, organiziranje podataka u hash strukture itd. Pripremljeni podatci dani su kao ulaz u klasifikator pravila pridruživanja. Konačni korak je otkrivanje uzorka izvedenog različitim metodama grupiranja. U [9] korišteno je grupiranje temeljeno na optimizaciji kolonijom mrava, algoritam za podjelu grafova i fuzzy grupiranje. Na temelju dobivenih rezultata, fuzzy grupiranje je bila najbolja opcija za predviđanje navigacijskog uzorka.

Grupiranje web korisnika uz pomoć fazi k-sredina algoritma predstavljeni su u [4]. Predložena metoda omogućuje da jedna instanca bude dio ne samo jedne grupe, već više njih. Osim određivanje grupa za jedan primjer, izračunava se i stupanj njegove pripadnosti. Rezultati simulacije pokazali su da je predložena metoda brža i uspješnija za veći broj grupa.

Metoda grupiranja prilagodljivog praga za raščlanjivanje imena osoba predložena je u [32]. Kvaliteta grupiranja uvelike ovisi o odabranim pragovima. Obično se određuju uz pomoću skupa trening podataka koji nije uvijek moguć za web podatke. Predložena metoda ovisi samo o sadržaju dokumenata i dinamički prilagođava vrijednosti threshold. Testirana je na tri skupa podataka, a rezultati su uspoređeni s drugim najmodernijim metodama. Usporedba rezultata dovodi do zaključka da je predložena metoda imala najbolju izvedbu.

U ovoj disertaciji razmatrana je i testirana metoda za grupiranje podataka web inteligencije koja predstavlja kombinaciju algoritma inteligencije rojeva, ogoljenog algoritma vatrometa i algoritma k-sredina. U predloženoj metodi primjenjuju se koraci k-sredina i BBFWA alternativno. Napravljena je usporedba s drugim pristupima iz literature. Na temelju eksperimentalnih rezultata može se zaključiti da predložena

metoda ima vrlo obećavajuće karakteristike u pogledu kvalitete grupiranja, kao i vremena izvršavanja.

Budući da kvaliteta grupiranja algoritmom k-sredina ovisi o početnom odabiru položaja centroida, predstavnika svakog klastera, prilično se različita rješenja mogu dobiti u nekoliko pokretanja algoritma sa različitim inicijalnim odabirom klastera. Kako bi se pronašlo bolje rješenje, algoritam k-sredina može se pokrenuti nekoliko puta s različitim početnim centroidima, a najbolje pronađeno rješenje može se upotrijebiti kao konačno. To je ekvivalentno metodi slučajnog pretraživanja što je jedan od najprimitivnijih metoda za pronalaženje optimalnog rješenja. Daleko bolja opcija je upotreba nekog algoritma za optimizaciju s upravljačkim mehanizmom, kao što su algoritmi inteligencije rojeva. U ovoj disertaciji ispitana je kvaliteta ogoljenog algoritma vatrometa za rješavanje ovog problema.

U ovoj disertaciji kombiniran je algoritam k-sredina s ogoljenim algoritmom vatrometa (engl. bare bones firework algorithm, BBFWA) kako bi se poboljšala brzina konvergencije algoritma i poboljšala kvaliteta rješenja. Na početku se centroidi postavljaju kao slučajne točke u prostoru pretrage koji je definiran instancama iz skupa. Nakon toga se provodi jedna iteracija k-sredina. Instance se dodjeljuju klasterima čiji centroid im je najbliži (centroidi su slučajno odabrane točke). Centroidi se ažuriraju i koriste kao kandidatsko rješenje BBFWA u sljedećoj generaciji. Za svako generirano kandidatsko rješenje u jednoj generaciji provodi se jedna iteracija algoritma k-sredina. To predstavlja fino prilagođavanje rješenja koje se generira s BBFWA. Naposljetku, BBFWA bi se sam približio tom rješenju, ali uvođenjem jedne iteracije algoritma k-sredina, brzina ovog procesa značajno se poboljšava. Kao što je već spomenuto, rješenje k-sredina uvelike ovisi o početnom izboru centroida. Na ovaj je način, korištenjem raznih inicijalizacija, umjesto nasumičnog pretraživanja, korištena BBFWA s vođenim pretraživanjem.

Pseudokod predložene metode dan je u Algoritmu 8.

Algoritam koji je predložen u ovoj disertaciji implementiran je u Matlab programskom jeziku, verzija R2016b. Svi su eksperimenti izvedeni na Intel ® Core ™ i7-3770K CPU @ 4GHz, 8GB RAM računala s Windows 10 Professional operativnim sistemom.

Na početku predložena metoda testirana je na dva jednostavna skupa podataka

Algorithm 8 Pseudokod predloženog algoritma za grupiranje podataka web inteligencije

Require: podatci web inteligencije, broj grupa k, parameteri BBFWA algoritma C_a and C_r, broj iskri n, donje i gornje granice prostora pretraživanja Lb and Ub

Generirati $x \sim U(Lb, Ub)$
Prilagoditi x s jednom iteracijom algoritma k-sredina
Evaluirati $f(x)$
$A = Ub - Lb$
repeat
  for $i = 1$ do n **do**
    Generirati $s_i \sim U(x - A, x + A)$
    Primjeniti operator mapiranja na s_i
    Prilagoditi s_i s jednom iteracijom algoritma k-sredina
    Evaluirati $f(s_i)$
  end for
  if $\min_{i=1,2,\ldots,n}(f(s_i)) < f(x)$ **then**
    $x = \text{argmin}(f(s_i))$
    $A = C_a A$
  else
    $A = C_r A$
  end if
until nije zadovoljen kriterij zaustavljanja

return x

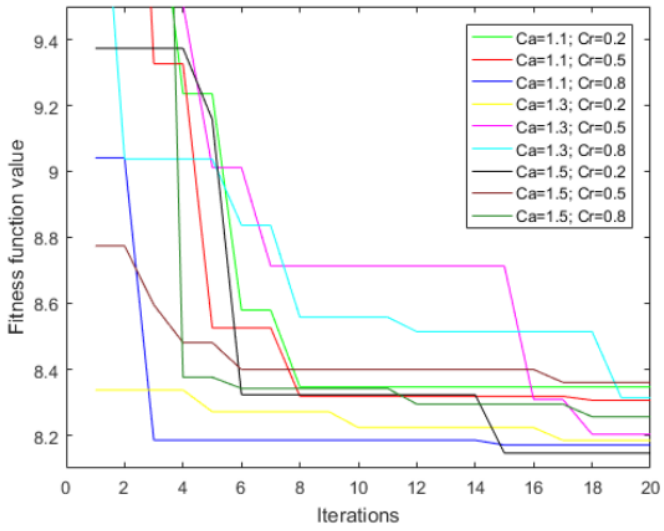
gdje su instance opisane s dva atributa. Ova karakteristika ih čini pogodnim za grafički prikaz i jednostavnu analizu algoritama grupiranja.

Prvi skup podataka zove se *Mouse* koji se može naći u različitim repozitorijima za rudarenje podataka i strojno učenje, poput *ELKI data mining* [95]. *Mouse* skup podataka sadrži 500 dvodimenzionalnih instanci koje je potrebno podijeliti u tri grupe. Drugi skup podataka je *Robot* (puni naziv skupa podataka je *Wall-Following Robot Navigation Data*) koji se može besplatno preuzeti u UCI Machine Learning Repository [69]. *Robot* je skup podataka koji sadrži 5456 instanci koje bi trebalo podijeliti u 4 klastera.

Skupovi podataka *Mouse* i *Robot* su korišteni za podešavanje BBFWA parametara C_a i C_r kao i za podešavanje maksimalnog broja evaluacija funkcije cilja.

Maksimalni broj evaluacija fitnes funkcije postavljen je na 400. Za potrebe testiranja, broj iskri odnosno rješenja u jednoj iteraciji n postavljen je na 20, dok je maksimalan broj iteracija postavljen također na 20. Ovim je rješenje dobiveno poslije

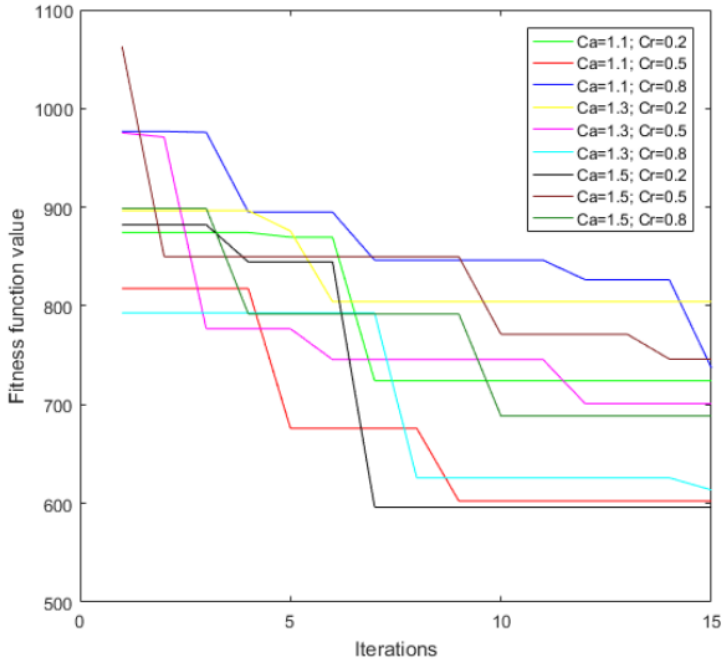
400 evaluacija funkcije cilja što je zbroj udaljenosti instanci od odgovarajućih centroida definiran jednadžbom **??**. Prvi zadatak je bio postaviti optimalne vrijednosti za parametre C_a i C_r. Proces podešavanja parametara odnosno minimalne vrijednosti fitnes funkcije u svakoj generaciji za nekoliko parova vrijednosti C_a i C_r prikazane su na slici 13. Na ovom grafu može se vidjeti da za sve parove parametara predloženi algoritam ima brzu konvergenciju, što se može objasniti kao posljedica jednostavnosti skupa podataka *Mouse*. Najbolji rezultati dobiveni su za par $(C_a, C_r) = (1.5, 0.2)$.



Slika 13: *Konvergencija predloženog algoritma primijenjenog na skup podataka Mouse*

Isti test za određivanje parametara izveden je i na *Robot* skup podataka. Rezultati su prikazani na slici 14. Može se vidjeti da su najbolji rezultati postignuti istim parom vrijednosti za C_a i C_r kao za skup podataka *Mouse*. Na osnovu ova dva rezultata, parametri C_a i C_r su postavljeni na sljedeće vrijednosti $C_a = 1.5$ i $C_r = 0.2$

Budući da ova dva skupa podataka mogu biti grafički predstavljeni, upotrijebljeni su za usporedbu algoritama čiste k-sredine s predloženom metodom u ovom radu. Na slici 15 grupirani su podatci *Mouse* s algoritmom k-sredina i s predloženim algoritmom s parametrima $C_a = 1.5$ i $C_r = 0.2$. Budući da je ovaj skup podataka prilično jednostavan, oba algoritma su ustanovila da minimalna vrijednost fitnes funkcije iznosi 8.1132, a centroidi su u oba slučaja na istom mjestu:



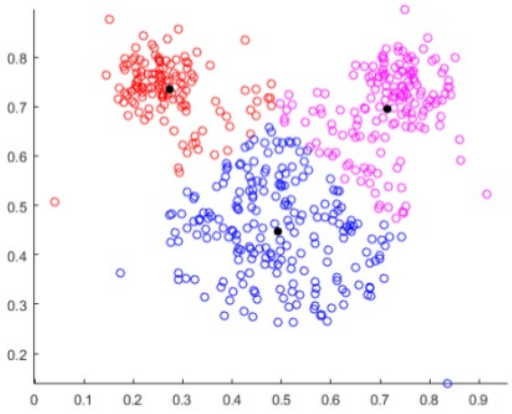
Slika 14: *Konvergencija predloženog algoritma primijenjenog na skup podataka Robot*

$$Centroids = \begin{pmatrix} 0.4925 & 0.4479 \\ 0.2731 & 0.7342 \\ 0.7139 & 0.6944 \end{pmatrix}$$

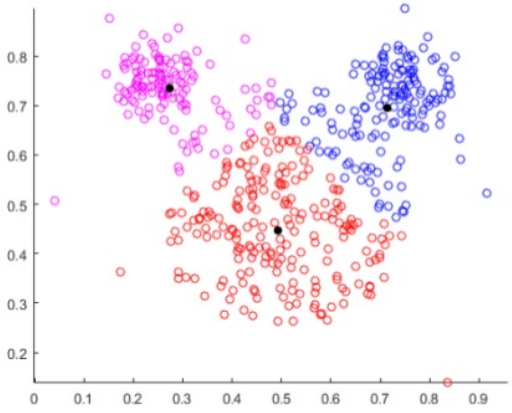
Razlika između izvornog algoritma k-sredina i predloženog BBFWAu ovom radu, na temelju grupiranja, jasno je vidljiva na skupovima podataka *Robot*. Rezultati oba algoritma prikazani su na slici 16.

Algoritam k-sredina utvrdio je da je bolje staviti u jednu grupu samo nekoliko instanci koje su daleko od ostalih (zelena grupa na slici 16(a)). Posljedica toga je da se u drugu grupu (crvena na slici 16(a)) mora uključivati više instanci koje su prilično udaljene jedna od druge. S druge strane, predložena metoda je te udaljene i izdvojene instance grupirala zajedno s najbližom većom grupom (roza grupa na slici 16(b)). Zbog te grupacije, bilo je moguće grupirati veći broj udaljenih točaka u jedan klaster i time umanjiti vrijednost funkcije cilja (zelena grupa na slici 16(b)).

Algoritam k-sredina postiže 846.4235 za vrijednost funkcije fitnes i koordinate



(a)



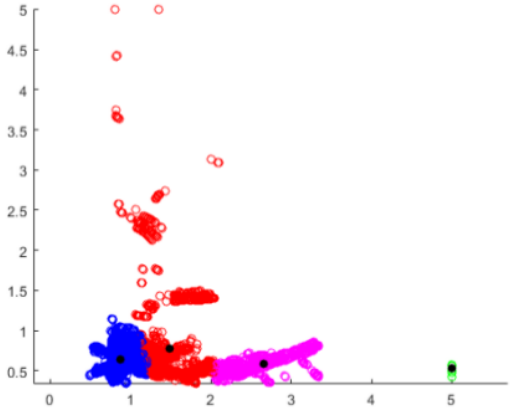
(b)

Slika 15: *Grupiranje skupa podataka Mouse s (a) algoritmom k-sredina i (b) predloženom metodom u ovom radu*

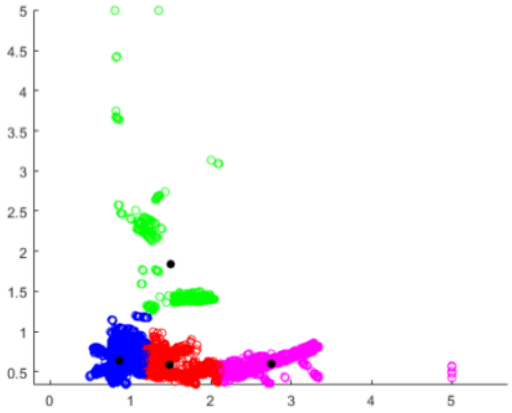
centroida su:

$$Centroids = \begin{pmatrix} 1.4907 & 0.7789 \\ 5.0000 & 0.5301 \\ 2.6633 & 0.5899 \\ 0.8689 & 0.6389 \end{pmatrix}$$

Minimalna vrijednost fitnes funkcije dobivene predloženom metodom u ovom radu je 538.4042 s koordinatama centroida:



(a)



(b)

Slika 16: *Grupiranje skupa podataka Robot s (a) algoritmom k-sredina i (b) predloženom metodom u ovom radu*

$$Centroids = \begin{pmatrix} 0.8642 & 0.6451 \\ 2.7483 & 0.5952 \\ 1.4950 & 0.5815 \\ 1.5064 & 1.8363 \end{pmatrix}$$

Kvaliteta predložene metode za grupiranje podataka web inteligencije testirana je na dva referentna (benchmark) skupa podataka iz UCI Machine Learning Repository [69]. Dva skupa podataka korištena u ovom radu korištena su također i u [92]. Prvi skup podataka *Page block* sadrži 5473 instanci, blokova izgleda web stranice iz 54

različita dokumenta. Ovaj skup podataka poslužio je kao benchmark za klasifikaciju i grupiranje web stranica. Svaku instancu opisuje 10 atributa, kao što su visina i duljina bloka, postotak crnih piksela u bloku, itd.

Drugi skup podataka koji je korišten za testiranje predložene metode je *Spambase*. Otkrivanje neželjene pošte (eng. spam) i razlikovanje neželjene pošte od neželjene web stranice, mailova, oglasa i još mnogo toga, predstavlja prilično važan zadatak web inteligencije. *Spambase* skup podataka sadrži 4601 instancu, tj. neželjenu i ne-neželjenu e-poštu prikupljenu s osobne i radne adrese. Sakupljene e-poruke opisane su s 57 atributa, poput postotka riječi u e-pošti koje se podudaraju s nekim uobičajenim neželjenim riječima, ukupno, prosječno i najduže trajanje neprekidnih nizova velikih slova, itd. Primjere treba podijeliti u dvije kategorije: neželjena pošta ili ne-neželjena pošta. Ove informacije o korištenim skupovima podataka prikazane su u tabeli 4.

Tabela 4: *Informacije o korištenim skupovima podataka*

Skup podataka	**Instance**	**Atributi**	**Grupe**
Page Blocks	5473	10	5
Spambase	4601	57	2

Predložena metoda je uspoređena s drugim algoritmima za grupiranje podataka web inteligencije predloženih u [92]. Nekoliko algoritama inteligencije rojeva korišteno je za grupiranje u [92]: algoritam krijesnica (C-Firefly), pretraživanje kukavice (C-Cuckoo), algoritam šišmiša (C-Bat) i algoritam pretraživanja vuka (C-WSA). Dobiveni rezultati uspoređeni su s optimizacijom roja čestica (C-PSO) i s k-sredina. Ovi rezultati su također uključeni u ovaj rad. Usporedba rezultata za prvi skup podataka *Blokovi stranica* prikazana je u tabeli 5. Višestruko izvođenje predloženog algoritma daje uvijek istu vrijednost fitnes funkcije, onu navedenu u tabeli 5 koja prikazuje robusnost metode. Također, izvještavaju i o vremenima izvršenja.

Predložena metoda utemeljena na BBFWA za grupiranje podataka web inteligencije dobila je najbolju vrijednost fitnes funkcije, 1.3218E + 10. Algoritam pretraživanja kukavice grupirao je podatke na način da je vrijednost fitnes funkcije bila 1.3559E + 10 što je drugi najbolji rezultat. Ostali algoritmi su imali slabije rezultate. Vrijeme izvršenja za predloženi BBFWA iznosilo je 26.0951, što je gore od vremena za C-PSO, C-Bat i C-WSA. U eksperimentima veličina populacije i

Tabela 5: *Usporedba podataka za skup podataka Page Blocks*

Algoritam	Vrijednost fitnes funkcije	Vrijeme izvršenja (s)
K-Means	1.8429E+10	**1.1658**
C-PSO	1.8416E+10	22.38212
C-Firefly	1.8415E+10	160.6491
C-Cuckoo	1.3559E+10	41.2793
C-Bat	1.8477E+10	19.1616
C-WSA	1.8421E+10	21.4579
BBFWA	**1.3218E+10**	26.0951
BBFWA (300 eval.)	**1.3218E+10**	18.0229

maksimalni broj iteracija postavljen je na iste vrijednosti kao u [136]. Međutim, predložena metoda ima bržu konvergenciju i optimalna vrijednost fitnes funkcije pronađena je u manje od 300 evaluacija fitness funkcije za koje je bilo potrebno 18.0229s. Algoritam k-sredina ima značajno kraće vrijeme izvršenja, ali ne može naći optimalno rješenje.

Rezultati prijavljeni u [92] i rezultati dobiveni predloženim algoritmom u ovom radu na drugom skupu podataka *Spambase* prikazani su u tabeli 6. Predložena metoda u ovoj disertaciji je zajedno s C-Cuckoom pronašla najbolje grupiranje skupa podataka *Spambase* s vrijednošću fitnes funkcije 9.4347E + 08. Svi ostali algoritmi, osim C-Bat-a našli su slično rješenje, 9.4348E + 08, dok je C-Bat imao najgore rješenje, 9.4469E + 08. Algoritam pretraživanja kukavice imao je malo bolje vrijeme izvođenja, ali predloženi algoritam pronalazi rješenje u samo 10 iteracija za koje je potrebno 6.7154s. Ostali algoritmi, osim k-sredina, imali su lošija vremena izvršavanja.

Tabela 6: *Usporedba podataka za skup podataka Spambase*

Algoritam	Sr. vrijednost fitnes funkcije	Vrijeme izvršenja (s)
K-Means	9.4348E+08	**0.8815**
C-PSO	9.4348E+08	15.8362
C-Firefly	9.4348E+08	80.4213
C-Cuckoo	**9.4347E+08**	24.6962
C-Bat	9.4469E+08	12.6976
C-WSA	9.4348E+08	15.3639
BBFWA	**9.4347E+08**	12.9505
BBFWA (200 eval.)	**9.4347E+08**	6.7154

6.2 Rezultati prilagodjavanja algoritama za prostor cjelobrojnih rješenja

6.2.1 Segmentacija slike mozga zasnovana na algoritmu krijesnica u kombinaciji s algoritmom grupiranja K-sredina

U ovom poglavlju razmatrani problem segmentacije medicinskih slika mozga riješen je uz pomoć algoritma koji se temelji na kombinaciji algoritma k-sredina s algoritmom inteligencije rojeva, od kojih je odabran algoritam krijesnica. Segmentacija slika mozga provodi se s ciljem otkrivanja različitih anomalija poput glioma, metastatskog adenokarcinoma, metastatskog bronhogenog karcinoma i sarkoma. Algoritam krijesnice prilagođen je za razmatrani problem.Predložena metoda testirana je na nekoliko standardnih slika mozga iz Harvard Whole Brain Atlas [51], a rezultati su uspoređeni s drugim metodama iz literature od kojih ćemo neke spomenuti.

Algoritam k-sredina je jednostavna metoda grupiranja, ali s jednim nedostatkom. Određivanje optimalnih centroida predstavlja NP-teški optimizacijski problem, a kvaliteta konačnog rješenja određena je početnim položajima centroida. Najčešće korištena inicijalizacija je slučajna inicijalizacija, dok se u nekim aplikacijama početni centroidi mogu odrediti korištenjem različitih strategija. Jedna od strategija, s kojom su dobiveni veoma uspješni rezultati, je korištenje različitih algoritmima inteligencije rojeva.

Za pronalaženje optimalnih centroida korišten je algoritam krijesnice. Budući da rezultat algoritma k-sredina ovisi o početnom rješenju, ali ima dobru konvergenciju, za traženje rješenja koristili smo algoritam krijesnica na način da su dobivena rješenja poboljšana u svakoj generaciji jednom iteracijom algoritma k-sredina.

Fitnes funkcija za algoritam krijesnice korištena u ovom radu je Otsuov kriterij koji se koristi za pronalaženje optimalnih graničnih vrijednosti na temelju histograma slike. Otsuova metoda često se koristila za segmentaciju slike kao na primjer u [97] i [129]). Kvaliteta grupiranja ili segmentacija određuje se ukupnom udaljenošću između grupa. Cilj je maksimizirati tu udaljenost koja je jednaka minimizaciji udaljenosti unutar grupe. Ako $h(i)$ predstavlja broj piksela s intenzitetom i na slici sive razine L, tada je Otsuov kriterij definiran sljedećom jednadžbom:

$$f(t) = \sum_{i=0}^{K} \omega_i \sigma_i^2 \tag{42}$$

gdje je

$$N = \sum_{i=0}^{L-1} h(i) \tag{43}$$

$$p(i) = \frac{h(i)}{N}, \;\; 0 \leq i \leq L-1 \tag{44}$$

i

$$\omega_i = \sum_{j=t_i}^{t_{i+1}-1} p(i) \tag{45}$$

$$u_i = \sum_{j=t_i}^{t_{i+1}} j \cdot \frac{p(i)}{\omega_i} \tag{46}$$

$$\sigma_i = \sum_{j=t_i}^{t_{i+1}-1} (i - \omega_i)^2 \cdot \frac{p(i)}{\omega_i}. \tag{47}$$

U prethodnim jednadžbama smatralo se da je $t_0 = 0$, a $t_L = L - 1$. Da bi se uspio Otsuov kriterij iskoristiti kao fitnes funkcija, potrebno je pronaći prag vrijednostima (threshold) koje odgovaraju središtima grupa dobivenih kao kombinacija algoritama grupiranja k-sredina s FA. Nakon dodjeljivanja instanca u odgovarajuće grupe, threshold vrijednosti određuju se prema najvećoj razini intenziteta piksela unutar svake grupe. Dobivene vrijednosti se sortiraju i koriste za izračunavanje Otsu-ovog kriterija čije vrijednosti predstavljaju mjeru kvalitete rješenja dobivenih kombinacijom algoritmom FA i k-sredina.

Predložena metoda za segmentaciju slike predstavljena je u Algoritmu 9.

Algorithm 9 Pseudokod za predloženi KM-FA algoritam

```
Inicijalizacija
Metodom slučajnog izbora inicijalizirajte n rješenja
Odredite prag vrijednosti (threshold) za svako rješenje, sortirati vrijednosti i izračunati
vrijednosti fitnes funkcije
Izračunati kvalitetu svakoga rješenja
repeat
    for i = 1 to n do
        for j = 1 to i do
            if I_i < I_j then
                pomicanje rješenja x_i prema rješenju x_j koristeći jednadžbu
                Ažurirati rješenja kroz jednu iteraciju algoritma k-sredina
                Odrediti prag vrijednostima (threshold), sortirati vrijednosti i izračunati
                vrijednosti fitnes funkcije
            end if
            Izračunati atraktivnost za rješenja x_i i x_j;
        end for
    end for
    Rangirati rješenja i postaviti najboljeg za centar
until Maksimalni broj iteracija je postignut.
```

U sljedećem koraku koristi se još jedna primjena algoritama inteligencije rojeva na praktične probleme i to njihova najjednostavnija prilagodba.

Nakon dobivanja konačnih rješenja, tj. threshold vrijednosti, segmentacija se izvršava postavljanjem piksela iz jednog segmenta (eng. cluster) na donju granicu segmenta. Budući da mjera kvalitete korištene za performanse evaluacije koristi razlike u intenzitetima piksela između izvorne i segmentirane slike, izbor predstavnika segmenta je veoma važan.

Kvaliteta segmentiranih slika dobivenih kombiniranim algoritmom k-sredina i FA mjerena su istom metrikom koja se koristila u [83]. Korištene metrike su normalizirani korijen srednje kvadratne pogreške (eng. normalized root mean squared error, NRMSE), vršni omjer signala i šuma (eng. peak signal-to-noise ratio, PSNR) i indeks sličnosti struktura (eng. structural similarity index, SSIM).

NRMSE se izračunava sljedećom jednadžbom:

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{1} \sum_{j=1}^{M} (x(i,j) - x'(i,j))^2}{\sum_{i=1}^{N} \sum_{j=1}^{M} x(i,j)^2}} \tag{48}$$

gdje je slika NxM dok su $x(i,j)$ i $x'(i,j)$ intenziteti piksela na položaju (i,j) izvorne i segmentirane slike. Manje vrijednosti za NRSME su bolje, gdje za dvije identične

slike vrijednost NRSME je 0.

PSNR se mjeri u dB i izračunava se na sljedeći način:

$$PSNR = 10 \log_{10} \frac{L^2}{MSE}, \tag{49}$$

gdje je L najveća vrijednost intenziteta, a MSE (eng. mean squared error) srednja kvadratna pogreška i izračunava se na sljedeći način:

$$MSE = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} (x(i,j) - x'(i,j))^2 \tag{50}$$

Bolje vrijednosti za PSNR su veće vrijednosti, budući se postižu ako je vrijednost od MSE manja, što znači da dvije slike imaju manje razlika.

SSIM indeks sličnosti opisuje sličnost dviju slika, u našem slučaju između izvorne i segmentirane slike. SSIM uzima vrijednosti između -1 i 1 pri čemu dvije identične slike imaju vrijednost SSIM jednaku 1, dok dvije slike bez strukturne sličnosti imaju vrijednost SSIM jednaku 0. Izračunava se kao:

$$SSIM = \frac{(2\mu_n\mu_m + d_1)(2\sigma_{nm} + d_2)}{(\mu_n^2 + \mu_m^2 + d_1)(\sigma_n^2 + \sigma_m^2 + d_2)} \tag{51}$$

gdje μ_n i μ_m predstavljaju srednje vrijednosti izvorne slike m i segmentirane slike n, σ_{nm} je kovarijanca od m i n, dok su σ_m^2 iσ_n^2 varijance slike m, odnosno n. Varijable d_1 i d_2 koriste se za stabilizaciju podjela i ako L predstavlja vrijednost maksimalnog intenziteta onda:

$$d_1 = (0.01 \cdot L)^2 \tag{52}$$

$$d_2 = (0.03 \cdot L)^2 \tag{53}$$

Predloženi algoritam testiran je na platformi Intel ® Core™ i5-9700K CPU na 4GHz, 8GB RAM-a, Windows 10 Professional OS. Predloženi algoritam implementiran je pomoću Matlab verzije R2016a. Parametri algoritma krijesnice određeni su empirijski i njihove vrijednosti prikazane su u tabeli 7. Veličina populacije i maksimalni broj iteracija postavljeni su kao u Nanda i sur. [83] (2018.) tako da se može izvesti poštena usporedba.

Kvaliteta predložene metode uspoređena je s metodom koju su predložili Nanda

Tabela 7: *Inicijalni parametri za FA*

Inicijalni parametri	**Vrijednosti**
Parametar α	0.5
Privlačnost u r=0, β_0	0.2
Koeficijent apsorpcije γ	1.0
Veličina populacije	20
Maksimalni broj iteracija	100

i sur. [83]. Nanda i sur. predložili su metodu za otkrivanje tumora mozga koja se temelji na galaktičkoj optimizaciji rojeva (engl. galactic swarm optimization, GSO) s algoritmom grupiranja k-sredina. Prvo početno rješenje algoritma GSO bilo je rješenje dobiveno algoritmom k-sredina. Funkcija fitnes koja se koristi u ovom radu, Otsuov kriterij, također je korištena u Nanda i sur. [83]. S ciljem poštene usporedbe rezultata, korištene su iste slike kao u Nanda i sur. [83].

Korištene slike dobivene su iz Harvard Whole Brain Atlas [51]. Slike koje su korištene su fdg-PET, titc-SPECT i MRI slike. Glioma je bio prisutan u fdg-PET i titc-SPECT slikama, dok MRI slike sadrže metastatski adenokarcinom, metastatski bronhogeni karcinom i sarkom. Sve su slike 8-bitne sive slike veličine 256x256. Broj segmenata odabran je na način da se istaknu tumori i omogući lakše otkrivanje u daljnjoj analizi. Na temelju karakteristika svakoga razmatranoga tumora i svake razmatrane slike broj grupa se može odrediti empirijski i u ovom radu korištene su standardne vrijednosti koje su također koristili Nanda i sur. [83]. U tabeli 8 prikazani su rezultati koji su prezentirali Nanda i sur. [83] zajedno s rezultatima dobivenim predloženim algoritmom k-sredina s algoritmom krijesnice (KM-FA). Najbolji rezultati ispisuni se podebljano.

Na temelju rezultata prikazanih u tabeli 8, može se zaključiti da je predložena KA-FA metoda postigla bolje rezultate u usporedbi s metodom koju su predložili Nanda i sur. i sve ostale metode koje se koriste za usporedbu, tj. orginalni GSO, stvarni kodirani genetski algoritam (RGA) i osnovni algoritam k-sredina. Nanda i sur. [83] predložili su KM-GSO metodu koja je postigla bolje rezultate za sve testne slike, osim za otkrivanje sarkoma gdje je pravi kodirani genetski algoritam pronašao bolje vrijednosti za NRSME i PSNR, dok je SSIM bio najbolji kada se koristio izvorni GSO. U usporedbi s metodom koju su predložili u [83], KM-GSO, predloženi algoritam

Tabela 8: *Usporedba između KM-FA algoritma i pristupa od Nanda i suradnika*

Slika	Algoritam	NRMSE	PSNR	SSIM
Glioma(fdg)PET	K-Means	0.6937	16.3426	0.5500
	GSO	0.6807	16.4458	0.5676
	RGA	0.6840	16.4580	0.5598
	KM-GSO	0.6782	16.4779	0.5676
	KM-FA	**0.4617**	**19.8192**	**0.6469**
Glioma(titc)SPECT	K-Means	0.3990	18.0223	0.6813
	GSO	0.3576	18.3697	0.6918
	RGA	0.3782	18.0892	0.6910
	KM-GSO	0.3537	18.4652	0.6950
	KM-FA	**0.3164**	**19.4321**	**0.7068**
Metastatic Adenocarcinoma	K-Means	0.2425	24.4619	0.8085
	GSO	0.3650	20.6039	0.7502
	RGA	0.3360	21.3050	0.8080
	KM-GSO	0.2410	24.4834	0.8164
	KM-FA	**0.1559**	**27.9912**	**0.8541**
Metastatic bronchogenic carcinoma	K-Means	0.3834	18.8560	0.6717
	GSO	0.3869	18.9737	0.6996
	RGA	0.3860	18.8488	0.6780
	KM-GSO	0.3388	20.1285	0.7146
	KM-FA	**0.2939**	**21.3631**	**0.7535**
Sarcoma	K-Means	0.5967	15.5893	0.6142
	GSO	0.5787	15.8206	0.6226
	RGA	0.5780	15.8233	0.6180
	KM-GSO	0.5863	15.7074	0.6220
	KM-FA	**0.4325**	**18.3508**	**0.6766**

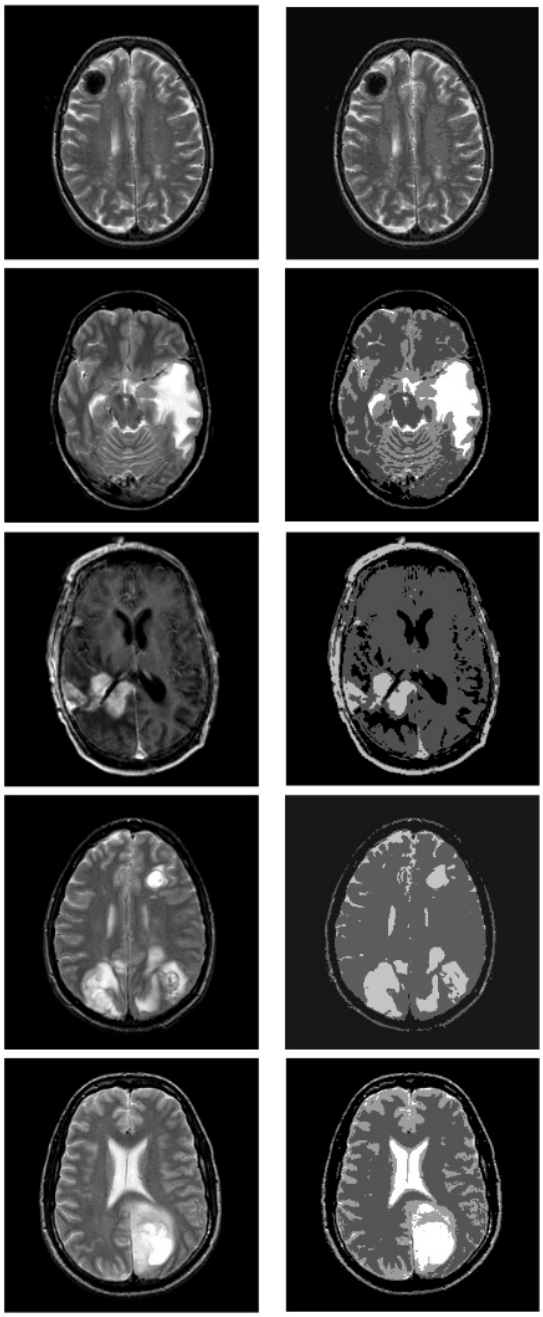
postigao je najbolje poboljšanje za gliom u ftg-PET slici. Za gliom na ftg-PET slici, KM-GSO, postupak koji su predložili u [83], dobio je NRMSE 0.6782 dok je predloženi KM-FA smanjio pogrešku na 0,4617. Najbolja vrijednost PSNR-a u Nanda i sur.[83] 16.4779 dobiveno je KM-GSO, dok je KM-FA metoda postigla 19.8192. SSIM dobiven KM-GSO bio je 0.5676, a predložena kombinirana metoda, KM-FA, dobila je SSIM jednak 0.6469. S druge strane, postignuto je najmanje poboljšanje za segmentaciju SPETC gliom slike kada su određena 4 klastera. NRMSE je bio 0.3164, što je nešto bolje od 0.3537, dobiveno segmentacijom KM-GSO. Poboljšanja u PSNR bila su od 18.4652 do 19.4321, dok je SSIM dobijen KM-GSO 0.6950, a SSIM po predloženom KM-FA 0.7068. Segmentacija metastatskog adenokarcinoma bio je problem optimizacije najveće dimenzije, to jest potrebno je utvrditi 8 klastera. U ovom slučaju je predložena metoda KM-FA dobila NRMSE, PSNR i SSIM 0.1559,

27.9912 i 0.8541, dok su ta mjerenja kvalitete za KM-GSO metodu u [83] iznosila 0.2410, 24.4834 i 0.8164.

Budući da je za segmentaciju korišten različit broj klastera, može se zaključiti da je predloženi algoritam KM-FA postigao dobre rezultate za obje, male i velike dimenzije razmatranog problema. Na temelju dobivenih slika, osim brojčanih rezultata, možemo vidjeti da je svaki razmatrani tumor uspješno naglašen dobivenom segmentacijom. Korištene slike i segmentirane slike dobivene predloženom metodom prikazane su na slici 17.

Algoritmi za digitalnu obradu slika za medicinske primjene izrazito su proučavano područje i potreba za preciznijim i bržim algoritmima je velika. U ovom poglavlju prikazana je metoda za segmentaciju slike mozga s ciljem otkrivanja različitih primarnih tumora. Slike mozga segmentiraju se algoritmom krijesnica kombiniranim s metodom grupiranja k-sredina kako bi se naglasile anomalije poput glioma, metastatskog adenokarcinoma, metastatskog bronhogenog karcinoma i sarkoma. Predložena metoda testirana je na standardnim referentnim slikama i dobila je bolje rezultate u usporedbi s drugim najmodernijim metodama iz literature.

Budući rad može uključivati automatsko određivanje broja grupa i prilagođavanje fitnes funkcije, uključujući prostorne informacije u procesu segmentacije.



Slika 17: *Originalne slike i njihove segmentirane slike dobivene našom predloženom metodom KM-FA algoritmom*

6.3 Rezultati prilagodjavanja algoritama za različite strukture rješenja

6.3.1 Problem planiranja putanje bespilotnih letjelica prilagođenim algoritmom krda slonova

Za rješavanje problema planiranja putanje UAV primjenjen je jedan od novijih algoritama inteligencije rojeva, algoritam za optimizaciju krda slonova (EHO). EHO se prije nije primjenjivao u polju planiranja putanje UAV-a i za ovaj problem predlažena je prilagođena verzija (AEHO) izvornog EHO algoritma. Nakon kratkog objašnjenja izvornog EHO algoritma, uvedena su pravila izvodljivosti, kao i detalji implementacije prilagođene verzije EHO algoritma za problem planiranja putanje UAV.

Algoritam za optimizaciju krda slonova predložen je 2016. godine od strane Wang i sur. [122]. Ponašanje u krdu slonova nadahnulo je autore na stvaranje novog obećavajućeg algoritma inteligencije rojeva. Složeno ponašanje u krdu pojednostavljeno je kako bi se implementirala eksploracija i eksploatacija u algoritam inteligencije rojeva. Populacija slonova sastoji se od nekoliko krda, a slonovi u krdima žive pod vodstvom matrijarha. Matrijarh je slon s najboljom fitnes vrijednošću u generaciji. Svaki član krda kreće se prema matrijarhu. U svakoj generaciji stalan broj slonova napušta krdo i odlazi živjeti daleko. Dakle, zaključak je da ponašanje u klanovima predstavlja eksploraciju, a napuštanje članova povezano je s procesom eksploatacije.

Izvorni EHO nije dizajniran za suočavanje s ograničenjima. Planiranje putanje UAV višestruko je ograničen, stoga je predlažen način postupanja s ograničenjima koji se temelji na pravilima izvodljivosti. Prije nego što se iskažu navedena pravila, potrebno je definirati neke pojmove koji se na njih odnose. Svaki agent (slon ili rješenje) koji predstavlja putanju UAV-a bit će kažnjen ako proizvoljna pod-putanja od putanje krši bilo koji od tri restriktivna uvjeta navedena prije. Zahvaljujući tome, rješenje će biti izvedivo. Znači, UAV se može sigurno kretati putanjom ako je ukupna penalizacija jednaka nuli. Inače, rješenje je neizvedivo. Za rješavanje ograničenja koristimo Debova pravila:

- Svako izvedivo rješenje je poželjnije od bilo kojeg neizvedivog;

- Među dva izvediva rješenja poželjnije je ono koje ima bolju fitnes vrijednost;
- Ako su oba rješenja neizvediva, preferira se ono s nižom kaznom.

Pojedinosti vezane za prilagođavanje i implementaciju EHO algoritma za rješavanje problema planiranja putanje UAV detaljno su raspisane po koracima.

Korak 1. *Generiranje inicijalne populacije*: Pretvoriti stari **XOY** koordinatni sustav u novi **X'O'Y '** kako je opisano prije. Generirati nasumično raspodijeljenu početnu populaciju $\mathbf{x}_i$ od n rješenja (slonovi) u novom **X'O'Y '** sustavu. Podijeliti cijelu populaciju slonova u k klanova. Procijeniti fitnes vrijednosti svih rješenja u skladu s funkcijom troškova definiranom u Eq. 39 i postaviti varijablu *cycle* na jedan. Prije pokretanja iterativnog postupka pretraživanja, sortirati populaciju zbog pravila izvodljivosti od najprikladnijeg $\mathbf{x}_{best}$ do najmanje odgovarajućeg $\mathbf{x}_{worst}$.

Korak 2. *Određivanje operatora za ažuriranje klana:* Za svako krdo c_i $(i = 1, 2, \cdots, k)$ u populaciji i za svakoga slona j $(j = 1, 2, \cdots, n/k)$ u krdu c_i, ažurirati staro rješenje $\mathbf{x}_{ci,j}$ i generirati novo $\mathbf{x}_{new,ci,j}$ po formuli:

$$\mathbf{x}_{new,ci,j} = \mathbf{x}_{ci,j} + \alpha \cdot (\mathbf{x}_{best,ci} - \mathbf{x}_{ci,j}) \cdot r \tag{54}$$

gdje $\mathbf{x}_{new,ci,j}$ označava novi položaj slona j u krdu i a $\mathbf{x}_{ci,j}$ je njegov stari položaj, $\mathbf{x}_{best,ci}$ je najbolje rješenje u krdu c_i, $\alpha \in [0, 1]$ je konstanta koja odeđuje utjecaj vodećeg slona matrijarha i $r \in [0, 1]$ je broj izvučen uniformnom raspodjelom iz intervala $[0, 1]$. U ovom koraku potrebno je provjeriti jednakost između rješenja $\mathbf{x}_{ci,j}$ i $\mathbf{x}_{best,ci}$. Ako su ta rješenja jednaka, ažurirati $\mathbf{x}_{ci,j}$ i generirati $\mathbf{x}_{new,ci,j}$ na sljedeći način:

$$\mathbf{x}_{new,ci,j} = \beta \cdot \mathbf{x}_{center,ci} \tag{55}$$

gdje je $\beta \in [0, 1]$ parametar algoritma koji kontrolira učinak rješenja $\mathbf{x}_{new,ci,j}$na nova rješenja $\mathbf{x}_{new,ci,j}$. Center $\mathbf{x}_{center,ci,j}$ krda c_i za dimenziju $d \in [0, D]$ izračuna se na sljedeći način:

$$\mathbf{x}_{center,ci,d} = \frac{1}{n_{ci}} \sum_{j=1}^{n_{ci}} \mathbf{x}_{ci,j,d} \tag{56}$$

gdje D predstavlja ukupnu dimenziju prostora i n_{ci} je broj slonova u krdu c_i. U ovom koraku računanja, potrebno je provjeriti granice novostvorenog rješenja $\mathbf{x}_{new,ci,j}$. Ako vrijednost bilo koje izračunate komponente rješenja $\mathbf{x}_{new,ci,j}$ prelazi dopuštene granice prostora za pretraživanje, tada će biti promjenjena u staru vrijednost.

Korak 3. *Izračun za operatora razdvajanja:* Odvajanje slonova može se koristiti za modeliranje procesa diverzifikacije. Dakle, u svakom krdu c_i populacije, najgori slon $\mathbf{x}_{worst,ci}$ koji napušta skupinu zamjenjuje slon koji se nasumično generira pomoću sljedeće formule:

$$\mathbf{x}_{worst,ci} = \mathbf{x}_{min} + (\mathbf{x}_{max} - \mathbf{x}_{min} + 1) \cdot rand \tag{57}$$

gdje parametar $rand \in [0, 1]$ je uniformno raspoređen broj, dok su parametri $\mathbf{x}_{min}$ i $\mathbf{x}_{max}$ donja i gornja granica prostora za pretraživanje.

Korak 4. *Zapamtiti najbolje trenutno rješenje:* Evaluirati fitnes vrijednosti populacije po novo ažuriranim pozicijama. Sortirati cijelu populaciju prema novim pravilima izvodljivosti, a zatim označiti najbolje rješenje kao $\mathbf{x}_{best}$ s najboljom fitnes vrijednošću y_{min}. Najbolji slon $\mathbf{x}_{best}$ prenosi se na sljedeću generaciju kao prvi slon $\mathbf{x}_0$.

Ako je ispunjen kriterij prekida ili je varijabla *ciklus* jednaka maksimalnom broju ponavljanja, algoritam se zaustavlja. U suprotnom, treba povećati varijablu *ciklus* za jedan i ići na korak 2.

Korak 5. *Kriterij zaustavljanja:* Ako je ispunjen kriterij prekida ili je varijabla *ciklus* jednaka maksimalnom broju ponavljanja, algoritam se zaustavlja. U suprotnom, treba povećati varijablu *ciklus* za jedan i ići na korak 2.

Korak 6. *Vizualizacija:* Transformirati koordinate iz novog **X'O'Y'** koordinatnog sustava u stari **XOY** sustav i izvršiti vizualizaciju podataka.

Svi testovi za predloženu AEHO metodu provedeni su na Intelovom ® Core ™i7-3770K CPU-u na 4GHz, 8GB RAM-a i Windows 10 Professional OS-u. Predložena AEHO metoda implementirana je u programskom jeziku *C#*. Da bi se dokazala izvedivost i učinkovitost predložene AEHO metode, napravljena je usporedna analiza s drugim najmodernijim algoritmima [125]. U ovoj eksperimentalnoj studiji, predloženi AEHO algoritam je uspoređivan s optimizacijom kolonijom mrava (ACO),

optimizacijom zasnovanom na biogeografiji (engl. bibliography based optimization, BBO), diferencijalnom evolucijom (DE), evolucijskom strategijom (ES), genetskim algoritmom (GA), optimizacijom roja čestica (PSO), stud genetskim algoritmom (SGA), algoritmom krijesnica (FA) i modificiranim algoritmom krijesnica (MFA). Veličina populacije u svim algoritmima, osim AEHO-a, postavljena je na $N = 30$ kao u [125], dok je za AEHO broj slonova postavljen na 50. Isti broj evaluacija objektivnih funkcija postavljen je za sve algoritme čime se osigurala pravedna usporedba između spomenutih algoritama. Na primjer, ako je broj evaluacija funkcija jednak 1500, 50 generacija je uzeto za sve algoritme, osim za EHO algoritam za koji je rezervirano 30 generacija.

Dodatni kontrolni parametri za sve algoritme u eksperimentima postavljeni su kao u [125]: **ACO**: početna vrijednost feromona $\tau_0 = 1E-6$, konstanta ažuriranja feromona $Q = 20$, istraživanje konstanta $q_0 = 1$, osjetljivost na feromon $\alpha = 1$, osjetljivost na vidljivost $\beta = 5$, globalne i lokalne brzine propadanja feromona postavljene su na 0,9 i 0,5; **BBO**: vjerojatnost promjene staništa postavljena je na 1, granica vjerojatnosti imigracije po genu odabrana je od $[0, 1]$, veličina koraka za numeričku integraciju vjerojatnosti postavljena je na jednu, maksimalna stopa imigracije i migracije za svaki otok je 1, vjerojatnost mutacije je jedan; **DE**: ponder faktora $F = 0,5$, crossover konstanta $C_r = 0,9$; **ES**: $\lambda = 10$ potomstva po generaciji, standardna devijacija $\sigma = 1$ za promjenu rješenja; **GA**: jednosmjeran crossover s vjerojatnošću križanja postavljen je na jedan, za strategiju odabira bira se odabir kotača ruleta i vjerojatnost mutacije postavljena je na 0,01; **PSO**: globalno učenje inercijalnog faktora je postavljeno na 0,3, a kognitivni i socijalni faktori (za interakciju u roju) postavljeni su na jedan; **SGA**: jednosmjeran crossover s vjerojatnošću križanja je jedan, dok je vjerojatnost mutacije postavljena na 0,01; **EHO**: broj klanova k postavljen je na 5, dok su parametri α i β postavljeni na 0,5 i 0,1.

U ovom radu, raspon pretraživanja je bio $[-20, 20]$ i razmatran je problem $d-$dimenzija. Zone opasnosti, početak i cilj postavljeni su kao u [125] i navedeni su u tabeli 9.

Za različite vrijednosti dimenzije d kao i za različite brojeve evaluacija objektivnih funkcija izračunati su najbolje, srednje i najgore vrijednosti u 100 izvođenja. Rezultati eksperimenta predstavljeni u ovom radu nisu normalizirani, tj. nisu smanjeni za 50,

Tabela 9: *Informacije o poznatim područjima prijetnje*

No.	Lokacija (km)	Radijus prijetnje (km)	Level prijetnje
1.	(45, 50)	10	2
2.	(12, 40)	10	10
3.	(32, 68)	8	1
4.	(36, 26)	12	2
5.	(55, 80)	9	3

kao što je učinjeno u članku [125]. Dobiveni rezultati predstavljeni su u tabeli 10 i tabeli 11.

Prema eksperimentalnim rezultatima iz tabele 10, može se primijetiti da je za sve slučajeve evaluacije funkcija cilja, osim u jednom, predloženi AEHO algoritam nadmašio ostatak algoritama uzimajući u obzir sve statističke parametre. Za samo 1500 evaluacija, izuzev algoritama FA, MFA i AEHO, možemo istaknuti da GA i SGA daju bolje rezultate u odnosu na ostale algoritme. Točno, GA daje malo bolje rezultate od SGA, dok DE i PSO stvaraju vrlo slične rezultate uzimajući u obzir sve statističke parametre. Situacija postaje nešto drugačija kada se povećava broj evaluacija. Konkretno, za 3000 evaluacija, DE algoritam dava bolje rezultate od sljedećih algoritama: ACO, ES, BBO, PSO, GA i SGA. U ovom i prethodnom slučaju ACO daje najgore rezultate. Međutim, kako se broj evaluacija povećava na 4000 ili 7500, primjećuje se da algoritam DE ostaje na vrhu, odmah iza algoritama kao što su FA, MFA i AEHO. Također, za razliku od prije, ACO nije najgori algoritam, to je sada ES algoritam.

Važno je primijetiti da povećanjem broja evaluacija, algoritmi DE, GA, SGA, FA, MFA i EHO zadržavaju tendenciju za poboljšanjem svojih rješenja. Na temelju rezultata prikazanih u tabeli 10, zaključuje se da algoritmi AEHO, MFA, FA, DE, SGA, GA i PSO proizvode sasvim prihvatljiva pod-optimalna izvediva rješenja, gdje AEHO stvara najbolje rezultate u tom smislu, robusnosti i kvalitete.

Prema zabilježenim podatcima u tabeli 11, dodatno se razmatra utjecaj dimenzije D na performanse algoritma AEHO kao i na druge populacijske optimizacijske metode za rješavanje problema planiranja puta UAV.

Tabela 10: *Komparacija dobivenih rezultata između osam algoritama preko 100 nezavisnih ciklusa za d=20*

Alg.	**Stat.**	**BROJ EVALUACIJA FUNKCIJA CILJA**			
		1500	**3000**	**4500**	**7500**
ACO	BEST	59.6276	61.5242	55.6381	59.7607
	WORST	68.7099	67.7404	67.4223	67.0678
	MEAN	66.2648	66.3500	66.1722	66.0444
BBO	BEST	54.8658	54.2616	55.6105	53.5209
	WORST	78.6806	78.2427	90.1797	77.6544
	MEAN	64.2076	63.4074	62.6978	61.9654
DE	BEST	53.6814	50.9439	50.7015	50.4829
	WORST	81.3392	69.4058	64.5560	58.5122
	MEAN	63.2645	57.3195	53.5255	52.4849
ES	BEST	57.7983	60.2329	59.8027	60.2540
	WORST	83.1927	83.1979	85.7277	96.0828
	MEAN	70.3152	70.4387	69.9485	70.1739
GA	BEST	51.6448	51.5294	51.2042	50.8781
	WORST	60.9773	61.1678	67.5637	57.4338
	MEAN	54.1142	53.7731	53.4671	52.6605
PSO	BEST	53.6012	52.3357	52.6165	52.9229
	WORST	83.1539	77.8806	78.3542	79.6341
	MEAN	59.9671	58.9057	58.5509	59.2143
SGA	BEST	51.7041	51.3520	50.9498	50.7839
	WORST	67.6318	61.6446	65.2145	66.2672
	MEAN	55.2389	53.7475	53.1680	52.5929
FA	BEST	51.4713	50.6577	50.5459	50.4753
	WORST	78.0425	79.3022	77.8480	76.3005
	MEAN	56.2034	54.3526	54.1809	52.2064
MFA	BEST	**50.7030**	50.5382	50.4857	50.4508
	WORST	54.6726	54.5749	54.9631	53.6783
	MEAN	51.9576	51.3048	50.9933	50.7025
AEHO	**BEST**	50.7145	**50.50409**	**50.44509**	**50.40952**
	WORST	**52.78959**	**51.19080**	**50.91982**	**50.72708**
	MEAN	**51.32330**	**50.78713**	**50.65216**	**50.54372**

Tabela 11: *Usporedba rezultata optimizacije povezanih sa statističkim parametrima za osam algoritama za različite vrijednosti dimenzije d preko 100 ciklusa i 6000 procjena funkcije*

D	Stat.	ALGORITMI									
		ACO	BBO	DE	ES	GA	PSO	SGA	FA	MFA	AEHO
5	BEST	61.3724	60.3302	54.3562	59.5895	55.2471	55.1667	55.6538	54.3585	54.3573	**50.3748**
	WORST	63.3199	171.5724	62.2083	112.2665	61.6013	66.0713	61.2006	65.7395	62.4186	**50.3908**
	MEAN	61.5151	72.7318	58.5962	58.7499	60.4747	59.9061	60.5013	58.7499	59.1673	**50.3774**
10	BEST	60.2281	52.9472	51.3950	57.4272	51.6068	52.2073	51.5489	51.3990	51.3966	**50.3753**
	WORST	68.1911	76.8270	56.7358	123.4605	60.1096	68.6221	56.1652	56.7095	53.7858	**50.5320**
	MEAN	61.9485	57.9650	53.1045	76.2868	52.5422	57.0411	52.2790	52.1801	51.5740	**50.4117**
15	BEST	58.5298	52.5569	50.6114	58.2547	50.8711	52.0969	50.8071	50.6172	50.6115	**50.3899**
	WORST	61.0084	90.3705	62.5808	103.8683	57.4472	87.3201	61.7962	94.2763	53.8319	**50.4624**
	MEAN	60.2554	59.5257	52.2783	71.8681	52.1880	58.3395	51.8910	52.8217	50.8967	**50.4117**
20	BEST	61.2445	52.9424	50.5188	60.7960	51.0695	52.3469	50.8392	50.4626	50.4552	**50.41371**
	WORST	67.0679	82.1981	66.6736	102.4090	61.9124	78.2524	56.7380	78.9142	52.0279	**50.79684**
	MEAN	66.2154	61.8556	52.3975	70.7501	52.9711	58.9892	52.3792	53.7327	50.7004	**50.55386**
25	BEST	61.5490	55.5286	50.5512	63.3685	51.2421	53.7378	51.2394	50.4908	50.4571	**50.4537**
	WORST	62.0733	80.3315	69.6664	83.9148	60.3977	78.1399	65.6967	66.4518	53.7043	**51.0589**
	MEAN	61.5674	64.7800	54.4081	72.7794	53.7814	60.2627	54.1567	53.9039	50.9987	**50.6949**
30	BEST	63.2299	56.6071	50.8987	65.7251	51.9218	53.2993	51.6165	50.6828	**50.5160**	50.5385
	WORST	64.7139	78.5885	74.1279	91.3024	62.7183	93.6950	64.7140	65.9757	58.3364	**51.6222**
	MEAN	63.9593	67.8746	59.9884	74.7757	55.0079	62.3847	54.5211	54.9621	51.3568	**50.9716**
40	BEST	69.7946	63.5504	54.5490	68.2314	52.2084	55.7367	52.6180	51.5225	**50.4506**	51.1236
	WORST	77.0641	90.7087	93.2604	96.4224	72.0688	84.7302	67.8669	86.6626	57.7236	**55.8737**
	MEAN	74.5754	74.8531	77.6201	80.2595	57.4927	64.8845	57.1100	57.8558	52.1978	**52.1321**

Kao što je prikazano u tabeli 11, za gotovo sve dimenzije D, osim za $D = 30$ i $D = 40$, predloženi AEHO pristup uvijek može pronaći najbolja izvodljiva rješenja u odnosu na druge tehnike unutar svake od svih 100 vožnja. S druge strane, oba algoritma ACO i ES za svaku dimenziju D generiraju najgora optimalna rješenja u usporedbi s ostalim metodama. Tabela 11 pokazuje da je ovaj predloženi AEHO pristup bio najučinkovitiji za dimenziju $D = 5$, dok su ostali algoritmi bili najbolji za ostale dimenzije. Na primjer, algoritam MFA bio je najučinkovitiji za dimenzije $D = 40$, dok su metode PSO, GA, SGA, ACO i BBO bile najučinkovitije za dimenziju $D = 15$. Također, algoritam FA je postigao najbolje pod-optimalne rezultate za $D = 20$ kao i algoritam DE, dok je metoda ES postigla najbolju vrijednost za dimenziju $D = 10$.

Razmatrajući dimenzije veće od 10, možemo zaključiti da su svi algoritmi koji sudjeluju dali neznatno lošiju kvalitetu rezultata isključujući MFA metodu. Stoga, povećanjem dimenzije $D = 20$ s faktorom 5 ili više, spomenuti algoritmi, osim MFA algoritma, nisu u mogućnosti otkriti najbolje izvediv put kao što je to bio slučaj za $D \leq 20$.

Vjerojatno je glavni razlog tome mali broj generacija i neadekvatna uspostavljena ravnoteža između eksploracija i eksploatacija. To će biti glavna motivacija za buduća istraživanja koja će se usredotočiti na postizanje točnijih rezultata. Gotovo sve metode dovele su do prihvatljivih, izvedivih rješenja međutim najbolji rezultati postignuti su predloženom AEHO metodom u smislu robusnosti i kvalitete.

Simulacijski eksperimenti su pokazali da predložen AEHO pristup može proizvesti prihvatljivu, izvedivu pod-optimalnu putanju za problem planiranja putanja UAV-a, a gotovo uvijek daje bolje rezultate u usporedbi s ACO, BBO, DE, ES, GA, PSO, SGA, FA i MFA algoritmima, uzimajući u obzir i točnost i, posebno, stabilnost algoritama. Stoga je predložena metoda nadmašila sve testirane algoritme uzimajući u obzir gotovo sve statističke parametre.

6.3.2 Prilagođeni algoritam umjetne kolonije pčela za trokut minimalne težine

U ovom istraživanju, istaknuti algoritam inteligencije rojeva, algoritam umjetne kolonije pčela prilagođen je za traženje podoptimalnih rješenja za postizanje brže konvergencije i smanjenja vremena CPU-a pri rješvanju problema pronalaženja trokuta minimalne težine.

Prilagođeni ABC algoritam (engl. adjusted artificial bee colony, AABC) na problem pronalaženja trokuta minimalne težine primenjen je kroz sljedeće faza. U početnoj fazi potreban je skup slučajnih rješenja gdje se dolazi do prvog problema - generiranje skupa slučajnih triangulacija. Tu se postavlja jedno netrivijalno pitanje kako triangulirati zadani skup točaka. Za sada je u literaturi poznato nekoliko algoritama za pronalaženje proizvoljnih triangulacija, ali u svrhu ovog rada prilagođen je pohlepni algoritam prikazan u [60]. Vjerojatno najvažnija primjedba ove faze je činjenica da se inicijalizacija početne populacije provodi drukčije nego na uobičajeni način. Započinje se sa svim pčelama vezanim samo za jedan izvor hrane. Za početno rješenje odabrali smo isto gore spomenuto pohlepno rješenje, koje se pokazalo najučinkovitijim za ovu svrhu. Nakon što se utvrdi početna populacija pčela, tj. agenata, može se provesti sljedeća faza algoritma, faza zaposlenih pčela.

Paralelno, sekvencijalno i stohastično penjanje uzbrdo, koje se temelji na metodi prevrtanja ivica (engl. edge flipping), obilježava ovu, a djelomično i sljedeću fazu algoritma.

Faza pčela promatrača ne razlikuje se mnogo od prethodne. Jedina je razlika što je dozvoljeno za rješenje da se pčele izviđači odmah mogu popeti na lokalni optimum, a sve ostale pčele čekaju. Odnosno, penjanje na brdo izvodi se na samo jednoj pčeli tj. odabranom rješenju dok se ne postigne lokalni optimum. Također, ovdje se vodi evidencija najboljih pronađenih rješenja do sada.

Sve zaposlene pčele na kraju će dostići svoj lokalni optimum i nakon što to učine, neće moći poboljšati svoje rješenje na isti način kao ranije. Drugim riječima, oni će biti zaglavljeni u lokalnom optimumu. Tu dolazi faza pčela izviđača. Tijekom ove faze, pčelama koje su postigle lokalni optimum omogućava se da neko vrijeme lutaju po prostoru za pretraživanje. To se omogućava tako što agent dopušta odabrati bilo koji od rubova okruženih konveksnim pravokutnikom, a ne samo one popravljive.

Na taj je način agentu omogućeno kretanje uzbrdo, kao i nizbrdo, tijekom svog istraživanja. Koliko dugo bi agent trebao lutati, tj. koliko rubova treba preokrenuti pčela izviđača, određuje neki prethodno definirani konstantni faktor.

U ovom eksperimentalnom istraživanju, predloženi AABC algoritam uspoređen je s dvije druge standardne metaheurističke tehnike, simulicija kaljenja (engl. simulated annealing, SA) i PSO. Kako u literaturi nema standardnih zbirki instanci za problem triangulacije minimialne težine, stvorena je zbirka od 5 slučajeva gdje se svaki od njih sastoji od najmanje 15 točaka i maksimalno 23 točke. Na tim referentnim vrijednostima uspoređivani su algoritam iscrpne pretrage i predložena AABC metoda. U svrhu provjere kvalitete dobivenih rješenja, kao i stabilnosti generiranih rješenja, također smo nasumično stvorili sedam slučajeva gdje se svaki od njih sastoji od 40 točaka. Svaka instanca se zove RI-k-i, gdje k označava veličinu $i-te$ instance, gdje je $i = 1, ..., 7$.

Točke su jednoliko raspoređene i za svaku točku (x, y), koordinate su $x, y \in [-333, 333]$. Za eksperimentalne simulacije slučajevi su prethodno obrađeni kako bi se osiguralo da se slučajne točke ne pojavljuju u ravnini. Kroz eksperimentalnu procjenu, ocjenjena je primjenjivost prilagođene ABC metaheuriste za problem triangulacije minimalne težine. Predložena AABC metoda implementirana je u programskom jeziku Python. Svi su testovi provedeni na Intel Core i7-3770K @ 3.5GHz s 16 GB RAM-a koji radi pod Windows 10 x 64 operativnim sustavom.

Da bi se osigurala usporedba predloženog AABC algoritma s PSO i SA algoritmima, izračunavanje ciljne funkcije izračunato je N $\times$ G puta, gdje je N broj populacije, a G je maksimalni broj generacija. Svaki je algoritam izvršen u 50 izvođenja. Veličina populacije za AABC postavljena je na N = 13 pčela (agenta), dok je za PSO postavljena N = 33 čestice. Vrijednost 'ograničenja' u fazi pčela izviđača AABC-a bila je jednaka 7. Parametri učenja algoritma PSO c_1 i c_2 inicijalizirani su na 2.0 i 2.0. Na početku algoritma PSO brzina čestica bila je postavljena na nulu. Broj generacija i za AABC i za PSO bio je postavljen na G = 33. U svrhu ovog rada, u slučaju algoritma simuliranog kaljenja, početna temperatura T_0 postavljena je na 1.0 s promjenom α ravnomjerno odabranom iz intervala (0.85, 0.99). SA algoritam se prekida nakon što ne promijeni težinu rješenja tijekom čitavog ciklusa.

Optimalne težine i odgovarajuća vremena računanja pronađena iscrpnim pre-

traživanjem, zajedno s odgovarajućim vremenima računanja za algoritam AABC, prikazani su u tabeli 12. Te vrijednosti se mogu usporediti s izračunatim najgorim, srednjim, najboljim i standardnim odstupanjima za 50 pokretanja za tri metaheuristička algoritma predstavljena u tabeli 13.

U tabeli 13 prikazani su samo redovi (referentni slučajevi) u kojima je postojala neka razlika u testiranim algoritamima. Za sve redove koji nisu prikazani u tabeli 13, svi algoritmi pronašli su najbolje rješenje u svim izvedbama. Kao što se vidi usporedbom tabele 12 i tabele 13, predloženi AABC stvorio je optimalnu težinu za 44 od 45 referentnih vrijednosti u samo 429 procjena.

Tabela 12: *Optimale težine i vrijeme računanja pronađene iscrpnim pretraživanjem i AABC algoritmom*

Slučajne Instance	**Iscrpna metoda pretraživanja**		**AABC**
	Optimalne težine	Vrijeme (s)	Srednje vrijeme (s)
RI-15-1	2111182	27.39	0.83
RI-15-2	1909236	16.26	0.75
RI-15-3	2049619	20.32	0.73
RI-15-4	1549159	19.40	0.79
RI-15-5	1995367	17.11	0.74
RI-16-1	1546841	59.25	0.74
RI-16-2	1702863	66.13	0.79
RI-16-3	1700495	59.36	0.78
RI-16-4	1463510	41.09	0.77
RI-16-5	2055712	32.89	0.87
RI-17-1	15762801	101.26	0.85
RI-17-2	1416507	206.64	0.87
RI-17-3	2326401	181.81	0.90
RI-17-4	1714096	131.35	0.87
RI-17-5	1550093	122.44	0.81
RI-18-1	2275043	240.71	1.07
RI-18-2	2696259	326.56	1.12
RI-18-3	2023911	408.3	1.07
RI-18-4	1806271	290.33	1.03
RI-18-5	1656804	147.41	1.02
RI-19-1	2006489	614.85	0.96
RI-19-2	1797201	401.56	1.14
RI-19-3	1777187	360.84	1.09
RI-19-4	1641380	263.18	1.04
RI-19-5	1685899	685.72	1.06
RI-20-1	2185726	709	1.09
RI-20-2	1537246	892.15	0.94
RI-20-3	1993018	1570.29	1.21
RI-20-4	1744241	1039.4	1.14
RI-20-5	2329993	1716.42	1.21
RI-21-1	2022009	912.2	1.27
RI-21-2	2473327	3226.62	1.25
RI-21-3	2113566	6737.32	1.51
RI-21-4	1381651	2281.67	1.25
RI-21-5	2348732	5508.41	1.26
RI-22-1	2246770	7446.92	1.25
RI-22-2	1820994	3626.2	1.38
RI-22-3	1755667	4961.5	1.33
RI-22-4	2068914	7846.59	1.37
RI-22-5	2291740	11868.93	1.37
RI-23-1	1644823	13573.4	1.44
RI-23-2	1890912	4254.22	1.34
RI-23-3	2515459	33332.45	1.46
RI-23-4	2347326	21125.1	1.36
RI-23-5	2252239	8613.89	1.46

Tabela 13: *Usporedba srednjih vrijednosti i standardnih odstupanja dobivenih za SA, PSO i AABC za pet slučajnih slučajeva tijekom 50 ciklusa.*

Rand.	**SA**			**PSO**				**AABC**			
Instan.	Mean	Best	SD	Worst	Mean	Best	SD	Worst	Mean	Best	SD
RI-15-1	2114203	2111182	1251	2111182	2111182	2111182	0	2111182	2111182	2111182	0
RI-15-3	2092213	2092213	0	2056895	2052665	2049619	3580	**2049619**	**2049619**	**2049619**	**0**
RI-15-5	2034210	1995367	10308	1995367	1995367	1995367	0	1995367	1995367	1995367	0
RI-16-1	1559225	1546841	2426	1546841	1546841	1546841	0	1546841	1546841	1546841	0
RI-16-3	1712957	1700495	3191	1701791	1700598	1700495	351	**1700495**	**1700495**	**1700495**	**0**
RI-17-2	1440159	1787137	4847	1416507	1416507	1416507	0	1416507	1416507	1416507	0
RI-17-3	2332225	2326401	1732	2326401	2326401	2326401	0	2326401	2326401	2326401	0
RI-17-4	1732083	1714096	3240	1714096	1714096	1714096	0	1714096	1714096	1714096	0
RI-18-2	2703399	2696259	3271	2696259	2696259	2696259	0	2696259	2696259	2696259	0
RI-18-5	1756330	1755344	491	1656804	1656804	1656804	0	1656804	1656804	1656804	0
RI-19-2	1804804	1797201	2920	1797201	1797201	1797201	0	1797201	1797201	1797201	0
RI-19-3	1781739	1777187	2156	1777187	1777187	1777187	0	1777187	1777187	1777187	0
RI-19-4	1645661	1641380	2054	1641380	1641380	1641380	0	1641380	1641380	1641380	0
RI-20-1	2248547	2185726	9824	2185726	2185726	2185726	0	2185726	2185726	2185726	0
RI-20-4	1756094	1744241	1659	1744241	1744241	1744241	0	1744241	1744241	1744241	0
RI-20-5	2343195	2329993	5468	2329993	2329993	2329993	0	2329993	2329993	2329993	0
RI-21-5	2357454	2348732	2366	2348732	2348732	2348732	0	2348732	2348732	2348732	0
RI-22-4	2081143	2068914	2672	2075512	2074265	2068914	2540	**2075512**	**2069309**	**2068914**	**1566**
RI-23-1	1652834	1644823	3204	1644823	1644823	1644823	0	1644823	1644823	1644823	0
RI-23-2	2285596	2252239	8243	2252239	2252239	2252239	0	2252239	2252239	2252239	0

Također, za sve slučajeve, AABC je uspio stvoriti optimalne težine u razumnom vremenskom roku, oko jedne sekunde, u usporedbi s iscrpnom metodom pretraživanja koja je u nekim slučajevima trajala više od 9 sati. Iz tabele 13 može se vidjeti da je SA u nekim slučajevima ostao zaglavljen u lokalnim optimumima, na primjer za slučajeve RI-15-3, RI-17-2 i RI-18-5. S druge strane, PSO i AABC su postigli zadovoljavajuće rezultate za svaki od 50 pokretanja. Ipak, AABC je daleko stabilniji, čak i uz činjenicu da je koristio manji broj evaluacija funkcije cilja. Budući da je PSO koristio 1089 evaluacija, to znači da je predloženi AABC više od dva puta brži od algoritma PSO. Stoga, prilagođeni algoritam ABC u svim slučajevima nadmašuje PSO i SA za sve testirane slučajeve. Uz to, dopuštajući ABC-u korištenje više sredstava, postoji mogućnost njegove još veće stabilizacije. Tabela 14 prikazuje testiranje na sedam većih instanci. Svaki od njih uključuje 40 točaka.

Predloženi AABC s 33 agenta ostao je potpuno stabilan (sa standardnim odstupanjem jednakim 0) za svako pokretanje i sve instance, dok se algoritam PSO snažno destabilizirao. Važno je naglasiti da je PSO zarobljen u nekim lokalnim optimumima, na primjer, RI-40-7, pa on nije sposoban postići globalni optimum.

Tabela 14: *Usporedba najgorih, srednjih, najboljih vrijednosti i standardnih odstupanja dobivenih u 50 ciklusa za SA, PSO i AABC za sedam slučajno generiranih slučajeva.*

Rand.	**SA**				**PSO**				**AABC**			
Instan.	Worst	Mean	Best	SD	Worst	Mean	Best	SD	Worst	Mean	Best	SD
RI-41-1	2849983	2848273	2847134	1395	2851232	2848588	2847134	1279	**2847134**	**2847134**	**2847134**	**0**
RI-42-2	2272714	2271293	2271062	573	2274818	2271969	2271062	942	**2271062**	**2271062**	**2271062**	**0**
RI-40-3	2385028	2384878	2384748	139	2401029	2390623	2384748	5666	**2384748**	**2384748**	**2384748**	**0**
RI-41-5	2390812	2384568	2382200	2964	2396415	2383177	2382200	2482	**2382200**	**2382200**	**2382200**	**0**
RI-42-4	3256158	3235455	3234082	5232	3234082	3234082	3234082	0	3234082	3234082	3234082	0
RI-43-6	2887767	2887106	2886920	350	2896889	2888788	2886920	2922	**2886920**	**2886920**	**2886920**	**0**
RI-40-7	2505964	2501009	2500908	707	2519285	2511475	2502858	4026	**2500908**	**2500908**	**2500908**	**0**

7 Zaključak

U ovoj doktorskoj disertaciji predloženo je prilagođavanje algoritama inteligencije rojeva na praktične probleme opzimizacije sa različitim prostorima pretrage. Područje kojim se ovaj rad bavi predstavlja važan problem rješavanja teških optimizacijskih problema i rad se oslanja na najnovija istraživanja u svijetu. Ovi metaheuristički algoritmi imaju jednostavnu implementaciju ali mogu učinkovito, za relativno kratko vreme, pronaći približna rješenja za NP-teške probleme optimizacije.

Kvalitet predloženih prilagođenih algoritama inteligencije rojeva je potvrđen korišćenjem metode komparativne analize sa metodama za rješevanje isih problema iz literature. Pri uporedbi, algoritmi su testirani pod istim uvjetima odnosno algoritmi su pokretani s istim brojem evaluacija funkcije cilja i nad istim test problemima. U toku znanstvenog i istraživačkog rada koristile su se različite metode kako bi bili zadovoljeni osnovni metodološki zahtjevi: objektivnost, pouzdanost, općenitost i sistematičnost. Metodologije istraživanja koje su se koristile u ovom radu uključuju metode kvantitativne i kvalitativne analize, statističke metode, metode analize sadržaja i metode komparativne analize.

U ovoj disertaciji razmatrane su sljedeće primjene algoritama inteligencije rojeva za rješavanje praktičnih problema sa različitim prostorima pretrage koji predstavljaju NP-teške probleme optimizacije:

1. u strojnom učenju za optimizaciju SVM
2. u strojnom učenju za minimizaciju grupiranja k-sredina
3. za segmentaciju digitalnih slika za dobivanje optimalnih treshold vrijednosti Otsuovom metodom
4. u određivanju optimalnih centroida
5. u problemu planiranja putanje bezpilotnih letjelica
6. u pronalaženju trokuta minimalne težine

U strojnom učenju algoritmi inteligencije rojeva koristili su se prilikom optimizacije SVM-a kao i u minimiziranju grupiranja algoritma k-sredina. Prostor pretrage za

ove probleme je $\mathbb{R}^D$ gde je D dimenzija problema.

Za optimiziranje modela stroja potpornih vektora potrebno je odrediti par parametara C i γ. U predloženoj metodi otkrivanja eritematoloških–skvamoznih bolesti pomoću optimizirane stroja potpornih vektora, parametri podešeni su prilagođenim algoritmom inteligencije rojeva, krdom slonova. Predložena metoda testirana je na standardnom skupu podataka koji sadrži podatke za 366 pacijenata s jednom od šest različitih eritematoloških–skvamoznih bolesti. Točnost predložene metode uspoređena je s točnošću drugih pristupa iz literature. Obavljena su četiri različita opita, a predložena metoda postigla je bolje rezultate u svim slučajevima.

Za grupiranje podataka web inteligencije predložen je algoritam vatrometa, i to njegova najjednostavnija verzija BBFWA, u kombinaciji s algoritmom k–sredina. Svako generirano rješenje pomoću BBFWA podešava se jednom iteracijom algoritma k-sredina. Uvođenjem k-sredina u algoritam optimizacije poboljšana je brzina konvergencije. Predložena metoda testirana je na benchmark skupu podataka za web iskopavanje. Dobiveni rezultati uspoređeni su s drugim pristupima iz literature koji za grupiranje koriste algoritme inteligencije rojeva. Iz literature, od algoritama inteligencije rojeva korišteni su optimizacija roja čestica, algoritam krijesnica, pretraživanje kukavice, algoritam šišmiša i algoritam pretraživanja vukova. Analizom rezultata može se zaključiti da predloženi BBFWA algoritam omogućava bolje grupiranje u smislu minimalne sume udaljenosti instanci od centroida grupa, kao i vremena izvršavanja. Daljnja istraživanja mogu uključivati strategiju određivanja inicijalnih centroida i dodatno prilagođavanje BBFWA.

U predloženoj metodi za segmentaciju slike mozga s ciljem otkrivanja različitih primarnih tumora korišćena je Otsuova metoda za određivanje trešhold vrednosti koje su cjelobrojne vrijednosti u intervalu $[0, 2^s - 1]$ gde je s broj bitova potreban za čuvanje vrijednosti jednog piksela. Za ovaj problem optimizacije bilo je dovoljno generirana realna rješenja zaokružiti na najbliži cijeli broj. Slike mozga segmentiraju se algoritmom krijesnica kombiniranim s metodom grupiranja k-sredina kako bi se naglasile anomalije poput glioma, metastatskog adenokarcinoma, metastatskog bronhogenog karcinoma i sarkoma. Predložena metoda testirana je na standardnim referentnim slikama i dobila je bolje rezultate u usporedbi s drugim metodama iz literature. Budući rad može uključivati automatsko određivanje broja grupa

i prilagođavanje funkcije cilja tako da uključuje prostorne informacije u procesu segmentacije.

Za riješavanje problema planiranja putanje bezpilotne letjelice u prostoru sa statičkim preprekama prilagođen je algoritam krda slonova i uspoređivan s drugim metodama prisutnim u literaturi. Prije primene algoritma inteligencije rojeva bilo je potrebno difnirati strukturu rješenja koja su predstavljena kao odstupanja od linije koja spaja početnu i krajnju točku putanje. Na taj način se bliskost dva rješenja uklapa u razmevanje bliskosti algoritma inteligencije rojeva. Predloženi algoritam testiran je na standardnim testovima iz literature, a pokazalo se da je superiorniji od svih ostalih korištenih algoritama. Štoviše, simulacijski eksperimenti pokazuju da je ovaj predloženi algoritam izvrstan izbor za problem planiranja puta bezbilotnih letjelica. Činjenica da prilagođeni algoritam krda slonova može pronaći bezbjednu putanju leta s najmanjim troškovima prijetnje i minimalnim troškovima goriva, postavlja EHO algoritam kao obećavajući pristup riješavanju ovog problema s mogućim daljnjim poboljšanjima kako bi se dobila preciznija rješenja.

Posljednji razmatrani problem u ovoj disertaciji je dizajn algoritma aproksimacije za rješavanje problema triangulacije minimalne težine za skupove točaka u dvodimenzionom prostoru. Za riješavanje ovog problema prilagođen je ABC algoritam i uspoređen je s drugim metaheuristikama. Ovaj predloženi prilagođeni ABC algoritam testiran je na 35 manjih i 7 većih instanci točaka u ravnini. Pokazalo se da je superiorniji od PSO, a posebno algoritma SA s obzirom na kvalitetu rješenja kao i na stabilnost dobivenih rješenja. To pokazuje da je naš predloženi algoritam odličan izbor za MWT problem. U budućnosti se mogu izvršiti dodatna podešavanja pomoću većih skupova točaka.

Buduća istraživanja mogu uključiti i brojna poboljšanja, konkretno inicijalne točke se mogu dobiti korištenjem određenih smjernica umjesto korištenje točaka slučajno raspoređenih u prostoru pretraživanja, prije korištenja predloženih metoda može se dodatno učiniti odabir obilježja, uspješnije uspostavljanje ravnoteže između eksploracije i eksplotacije, itd.

Opće je poznato da egzatne metode optimizacije garantiraju pronalazak optimalnog rješenja, međutim ne vode računa o pronalasku tog rješenja u razumnom vremenskom periodu. Za razliku od njih metaheurističke metode optimizacije prona-

laze rješenje koje se nalazi blizu optimalnog rješenja i to u razumnom vremenskom periodu. Međutim, zamjera im se da nemaju garanciju bliskosti optimuma za rješenje koje pronalaze. Sa sigurnošću možem tvrditi, da u ovom radu, to nije bio slučaj. Sve metaheurističke metode optimizacije, specijalno prilagođeni algoritmi inteligencije rojeva, koji su predloženi u ovom radu itekako su pogodni za primjenu na rješavanje praktičnih problema.

Na osnovu do sada navedenog mogu konkretno da se izdvoje znanstveni doprinosi ove disertacije:

1. *detaljan prikaz implementacije algoritama rojeva koje se nalaze u literaturi*
2. *pregled i detaljna analiza rezultata do kojih dolaze metaheuristike inteligencije rojeva*
3. *prilagođavanje postojećih implementacija algoritama inteligencije rojeva na konkretne NP-teške probleme optimizacije*
4. *istraživanje međusobnih odnosa i komparativna anliza algoritama inteligencije rojeva*
5. *algoritmi inteligencije rojeva uspješno su prilagođeni i primjenjivani za rješavanje praktičnih problema sa različitim prostorima pretrage*

Kvalitetu algoritama inteligencije rojeva primjenjenih na različite praktične probleme potvrđuju rezultati radova koji su objavljeni u svjetskim znanstvenim časopisima ili su izlagani na međunarodnim konferencijama čiji zbornici indeksirani u svetskim znanstvenim bazama.

8 Literatura

[1] Abdi, M.J., Giveki, D.: Automatic detection of erythemato-squamous diseases using PSO-SVM based on association rules. Engineering Applications of Artificial Intelligence 26(1), 603–608 (2013)

[2] Adeniyi, D., Wei, Z., Yongquan, Y.: Automated web usage data mining and recommendation system using k-nearest neighbor (KNN) classification method. Applied Computing and Informatics 12(1), 90–108 (2016)

[3] Agrawal, R., Sharma, M., Singh, B.K.: Segmentation of brain tumour based on clustering technique: Performance analysis. Journal of Intelligent Systems 28(2), 291–306 (2019)

[4] Ali, W., Alrabighi, M.: Web users clustering based on fuzzy c-means. VAWKUM Transactions on Computer Sciences 11(1), 1–9 (2016)

[5] Alihodzic, A.: Fireworks algorithm with new feasibility-rules in solving UAV path planning. In: The 2016 International Conference on Soft Computing and Machine Intelligence (ISCMI 2016). pp. 53–57 (November 2016)

[6] Alihodzic, A., Tuba, M.: Improved bat algorithm applied to multilevel image thresholding. The Scientific World Journal, special issue Computational Intelligence and Metaheuristic Algorithms with Applications, 2014(Article ID 176718), 16 (2014)

[7] Alihodzic, A., Tuba, M.: Improved hybridized bat algorithm for global numerical optimization. In: Proceedings of the 16th IEEE International Conference on Computer Modelling and Simulation, UKSim-AMSS 2014. pp. 57–62 (2014)

[8] Aljawawdeh, A., Imraiziq, E., Aljawawdeh, A.: Enhanced k-mean using evolutionary algorithms for melanoma detection and segmentation in skin images. Int. J. Adv. Comput. Sci. Appl. 8, 477–483 (2017)

[9] Anandhi, D., Ahmed, M.I.: Prediction of user?s type and navigation pattern using clustering and classification algorithms. Cluster Computing pp. 1–10 (2017)

[10] Andréasson, N., Evgrafov, A., Patriksson, M.: An introduction to optimization: Foundations and fundamental algorithms. Chalmers University of Technology Press: Gothenburg, Sweden 1, 1–205 (2005)

[11] Anter, A.M., Hassenian, A.E., Oliva, D.: An improved fast fuzzy c-means using crow search optimization algorithm for crop identification in agricultural. Expert Systems with Applications 118, 340–354 (2019)

[12] Arora, S., Barak, B.: Computational complexity: a modern approach. Cambridge University Press (2009)

[13] Arsuaga-Ríos, M., Vega-Rodríguez, M.A.: Multi-objective energy optimization in grid systems from a brain storming strategy. Soft Computing 19(11), 3159–3172 (2015)

[14] Aruna, S., Nandakishore, L., Rajagopalan, S.: A hybrid feature selection method based on IGSBFS and Naïve Bayes for the diagnosis of erythemato-squamous diseases. International Journal of Computer Applications 41(7) (2012)

[15] Asanambigai, V., Sasikala, J.: Adaptive chemical reaction based spatial fuzzy clustering for level set segmentation of medical images. Ain Shams Engineering Journal (2016)

[16] Astolfi, A.: Optimization – an introduction. Imperial Collage London pp. 1–80 (206), http://www3.imperial.ac.uk/pls/portallive/docs/1/7288263.PDF

[17] Bacanin, N., Tuba, M.: Fireworks algorithm applied to constrained portfolio optimization problem. In: IEEE Congress on Evolutionary Computation (CEC). pp. 1242–1249 (May 2015)

[18] Badrinath, N., Gopinath, G., Ravichandran, K., Soundhar, R.G.: Estimation of automatic detection of erythemato-squamous diseases through adaboost and its hybrid classifiers. Artificial Intelligence Review 45(4), 471–488 (2016)

[19] Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems, vol. 1. Oxford University Press (1999)

[20] Bouras, C., Tsogkas, V.: Improving news articles recommendations via user clustering. International Journal of Machine Learning and Cybernetics 8(1), 223–237 (2017)

[21] Cao, Z., Shi, Y., Rong, X., Liu, B., Du, Z., Yang, B.: Random Grouping Brain Storm Optimization Algorithm with a New Dynamically Changing Step Size, pp. 357–364. Springer International Publishing, Cham (2015)

[22] Caramia, M., Dell'Olmo, P.: Multi-objective Management in Freight Logistics. No. 1 in ncreasing Capacity, Service Level and Safety with Optimization Algorithms, Springer-Verlag London (November 2008)

[23] ling Chen, H., Yang, B., jing Wang, S., Wang, G., zhong Li, H., bin Liu, W., et al.: Towards an optimal support vector machine classifier using a parallel particle swarm optimization strategy. Applied Mathematics and Computation 239, 180–197 (2014)

[24] Chen, J., Cheng, S., Chen, Y., Xie, Y., Shi, Y.: Enhanced brain storm optimization algorithm for wireless sensor networks deployment. In: International Conference in Swarm Intelligence. pp. 373–381. Springer (2015)

[25] Chen, J., Wang, J., Cheng, S., Shi, Y.: Brain Storm Optimization with Agglomerative Hierarchical Clustering Analysis, pp. 115–122. Springer International Publishing, Cham (2016)

[26] Cheng, S., Qin, Q., Chen, J., Shi, Y.: Brain storm optimization algorithm: a review. Artificial Intelligence Review 46(4), 445–458 (2016)

[27] Chou, J.S., Cheng, M.Y., Wu, Y.W., Pham, A.D.: Optimizing parameters of support vector machine using fast messy genetic algorithm for dispute classification. Expert Systems with Applications 41(8), 3955–3964 (2014)

[28] Cobos, C., Muñoz-Collazos, H., Urbano-Muñoz, R., Mendoza, M., León, E., Herrera-Viedma, E.: Clustering of web search results based on the cuckoo search algorithm and balanced Bayesian information criterion. Information Sciences 281, 248–264 (2014)

[29] Coello, C.A.C.C.: A short tutorial on evolutionary multiobjective optimization. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 21–40. Springer (2001)

[30] Contreras-Cruz, M.A., Ayala-Ramirez, V., Hernandez-Belmonte, U.H.: Mobile robot path planning using artificial bee colony and evolutionary programming. Applied Soft Computing 30, 319–328 (2015)

[31] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT press (2009)

[32] Delgado, A.D., Martínez, R., Montalvo, S., Fresno, V.: Person name disambiguation in the web using adaptive threshold clustering. Journal of the Association for Information Science and Technology 68(7), 1751–1762 (2017)

[33] Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 26(1), 29–41 (1996)

[34] Dorigo, M., Caro, G.D., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial life 5(2), 137–172 (1999)

[35] Dorigo, M., Gambardella, L.M.: Ant colonies for the travelling salesman problem. Biosystems 43(2), 73–81 (1997)

[36] Dorzán, M.G., Leguizamón, M.G., Mezura-Montes, E., Hernández-Peñalver, G.: Approximated algorithms for the minimum dilation triangulation problem. Journal of Heuristics 20(2), 189–209 (2014)

[37] Dorzan, M., Gagliardi, E., Leguizamon, M., Hernandez Pealver, G.: Approximations on minimum weight triangulations and minimum weight pseudo-triangulations using ant colony optimization metaheuristic. Fundamenta Informaticae 119(1), 1–27 (2012)

[38] Duan, H.B., Zhang, X.Y., Wu, J., Ma, G.J.: Max-min adaptive ant colony optimization approach to multi-UAVs coordinated trajectory replanning in dynamic and uncertain environments. Journal of Bionic Engineering 6(2), 161–173 (2009)

[39] Fisher, L.: The perfect swarm: The science of complexity in everyday life. Basic Books (2009)

[40] Forsati, R., Moayedikia, A., Shamsfard, M.: An effective web page recommender using binary data clustering. Information Retrieval Journal 18(3), 167–214 (2015)

[41] Fu, Y., Ding, M., Zhou, C., Hu, H.: Route planning for unmanned aerial vehicle (uav) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems 43(6), 1451–1465 (2013)

[42] Gamare, P.S., Patil, G.: Web document clustering using hybrid approach in data mining. International Journal of Advent Technology 3(7), 92–97 (2015)

[43] Garey, M.R., Johnson, D.S.: A Guide to the Theory of NP-Completeness. San Francisco, Calif.: W. H. Freeman, Problem OPEN12 (1979), p. 228

[44] Graves, A., r. Mohamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 6645–6649 (2013)

[45] Gundes, P.: Optimization, `https://www.slideserve.com/derica/optimization`

[46] Havaei, M., Jodoin, P.M., Larochelle, H.: Efficient interactive brain tumor segmentation as within-brain kNN classification. In: 22nd International Conference on Pattern Recognition (ICPR). pp. 556–561. IEEE (2014)

[47] He, W., Mi, G., Tan, Y.: Parameter optimization of local-concentration model for spam detection by using fireworks algorithm. Advances in Swarm Intelligence, Springer LNCS 7928, 439–450 (2013)

[48] Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Tech. rep., National Taiwan University (2010)

[49] Hua, Z., Chen, J., Xie, Y.: Brain storm optimization with discrete particle swarm optimization for TSP. In: International Conference on Computational Intelligence and Security (CIS). pp. 190–193. IEEE (2016)

[50] Jia, Z., Duan, H., Shi, Y.: Hybrid brain storm optimisation and simulated annealing algorithm for continuous optimisation problems. International Journal of Bio-Inspired Computation 8(2), 109–121 (2016)

[51] Johnson, K.A., Becker, J.A.: The whole brain atlas [Online], http://www.med.harvard.edu/AANLIB/

[52] Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report - TR06 pp. 1–10 (2005)

[53] Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report - TR06, Erciyes University, Engineering Faculty, Computer Engineering Department pp. 1–10 (2005)

[54] Keller, E.F.: Organisms, machines, and thunderstorms: a history of self-organization, part two. complexity, emergence, and stable attractors. Hist Stud Nat Sci 39(1), 1–31 (2009)

[55] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. vol. 4, pp. 1942–1948 vol.4 (Nov 1995)

[56] Khalaf, E.T., Mohammad, M.N., Moorthy, K., Khalaf, A.T.: Efficient classifying and indexing for large iris database based on enhanced clustering method. Studies in Informatics and Control 27(2), 191–202 (2018)

[57] Kimura, A., Takahashi, I., Tanaka, M., Yasuda, N., Ueda, N., Yoshida, N.: Single-epoch supernova classification with deep convolutional neural networks. In: IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). pp. 354–359 (2017)

[58] Kolingerová, I., Ferko, A., Vomáčka, T., Maňák, M.: Nearest neighbour graph and locally minimal triangulation. In: International Conference on Computational Science and Its Applications. pp. 455–464. Springer (2017)

[59] Lee, E., Schmidt, M., Wright, J.: Improved and simplified inapproximability for k-means. Information Processing Letters 120, 40–43 (2017)

[60] Levcopoulos, C., Krznaric, D.: Quasi-greedy triangulations approximating the minimum weight triangulation. Journal of Algorithms 27(2), 303–338 (1998)

[61] Li, B., Gong, L.g., Yang, W.l.: An improved artificial bee colony algorithm based on balance-evolution strategy for unmanned combat aerial vehicle path planning. The Scientific World Journal 2014 (2014)

[62] Li, D., Cao, Y., Tang, X.s., Yan, S., Cai, X.: Leaf segmentation on dense plant point clouds with facet region growing. Sensors 18(11), 3625 (2018)

[63] Li, J., Tan, Y.: Enhancing interaction in the fireworks algorithm by dynamic resource allocation and fitness-based crowdedness-avoiding strategy. In: IEEE Congress on Evolutionary Computation (CEC). pp. 4015–4021 (July 2016)

[64] Li, J., Zheng, S., Tan, Y.: Adaptive fireworks algorithm. In: 2014 IEEE Congress on Evolutionary Computation. pp. 3214–3221. IEEE (July 2014)

[65] Li, J., Zheng, S., Tan, Y.: The effect of information utilization: Introducing a novel guiding spark in the fireworks algorithm. IEEE Transactions on Evolutionary Computation 21(1), 153–166 (Feb 2017)

[66] Li, J., Zheng, S., Tan, Y.: The effect of information utilization: Introducing a novel guiding spark in the fireworks algorithm. IEEE Transactions on Evolutionary Computation 21(1), 153–166 (Feb 2017)

[67] Li, J., Tan, Y.: The bare bones fireworks algorithm: A minimalist global optimizer. Applied Soft Computing 62, 454–462 (2018)

[68] Li, S., Sun, X., Xu, Y.: Particle swarm optimization for route planning of unmanned aerial vehicles. In: 2006 IEEE International Conference on Information Acquisition. pp. 1213–1218 (Aug 2006)

[69] Lichman, M.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2013), http://archive.ics.uci.edu/ml

[70] Lin, S.W., Ying, K.C., Chen, S.C., Lee, Z.J.: Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert systems with applications 35(4), 1817–1824 (2008)

[71] Lin, Y.B., Wang, R.M., Li, Y.: Minimum weight triangulation based on improved genetic algorithm with annealing selection. Journal of Information and Computational Science 1(2), 337–342 (2004)

[72] Liu, F., Zhou, Z.: A new data classification method based on chaotic particle swarm optimization and least square-support vector machine. Chemometrics and intelligent laboratory systems 147, 147–156 (2015)

[73] Liu, J., Qiao, S.: A image segmentation algorithm based on differential evolution particle swarm optimization fuzzy c-means clustering. Comput. Sci. Inf. Syst. 12(2), 873–893 (2015)

[74] Ma, L., Chen, H., Hu, K., Zhu, Y.: Hierarchical artificial bee colony algorithm for RFID network planning optimization. The Scientific World Journal 2014(Article ID 941532), 21 (2014)

[75] Marler, R.T., Arora, J.S.: The weighted sum method for multi-objective optimization: new insights. Structural and Multidisciplinary Optimization 41(6), 853–862 (June 2010)

[76] Mehrotra, S., Kohli, S.: The study of the usage of data analytic and clustering techniques for web elements. In: Proceedings of the ACM Symposium on Women in Research 2016. pp. 118–120. ACM (2016)

[77] Meng, L., Tan, A.H., Wunsch, D.C.: Adaptive scaling of cluster boundaries for large-scale social media data clustering. IEEE transactions on neural networks and learning systems 27(12), 2656–2669 (2016)

[78] Michalewicz, Z., Fogel, D.B.: How to solve it: modern heuristics. Springer Science & Business Media (2013), p. 554

[79] Moshref-Javadi, M.: Heuristic design and optimization, `http://www.mit.edu/~moshref/Heuristics.html`

[80] Mulzer, W., Rote, G.: Minimum-weight triangulation is NP-hard. Journal of the ACM (JACM) 55(2), 1–45 (2008), Article A11

[81] Muralidhar, A., Pattabiraman, V.: An efficient association rule based clustering of XML documents. Procedia Computer Science 50, 401–407 (2015)

[82] Mustaffa, Z., Yusof, Y., Kamaruddin, S.S.: Enhanced artificial bee colony for training least squares support vector machines in commodity price forecasting. Journal of Computational Science 5(2), 196–205 (2014)

[83] Nanda, S.J., Gulati, I., Chauhan, R., Modi, R., Dhaked, U.: A k-means-galactic swarm optimization-based clustering algorithm with otsu's entropy for brain tumor detection. Applied Artificial Intelligence 33(2), 152–170 (2019)

[84] Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization. Natural Computing Series, Springer-Verlag (November 2010)

[85] Olatunji, S.O., Arif, H.: Identification of erythemato-squamous skin diseases using extreme learning machine and artificial neural network. ICTACT Journal of Soft Computing 4(1), 627–32 (2013)

[86] Özçift, A., Gülten, A.: Genetic algorithm wrapped Bayesian network feature selection applied to differential diagnosis of erythemato-squamous diseases. Digital Signal Processing 23(1), 230–237 (2013)

[87] Ozturk, C., Hancer, E., Karaboga, D.: Improved clustering criterion for image clustering with artificial bee colony algorithm. Pattern Analysis and Applications 18(3), 587–599 (2015)

[88] Parasar, D., Rathod, V.R.: Particle swarm optimisation k-means clustering segmentation of foetus ultrasound image. International Journal of Signal and Imaging Systems Engineering 10(1-2), 95–103 (2017)

[89] Patil, R., Khan, A.: Bisecting k-means for clustering web log data. International Journal of Computer Applications 116(19) (2015)

[90] Qian, M., Zhai, C.: Unsupervised feature selection for multi-view clustering on text-image web news data. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management. pp. 1963–1966. ACM (2014)

[91] Ravichandran, K., Narayanamurthy, B., Ganapathy, G., Ravalli, S., Sindhura, J.: An efficient approach to an automatic detection of erythemato-squamous diseases. Neural Computing and Applications 25(1), 105–114 (2014)

[92] Rui, T., Fong, S., Yang, X.S., Deb, S.: Nature-inspired clustering algorithms for web intelligence data. In: IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT). vol. 3, pp. 147–153. IEEE (2012)

[93] Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, (2016)

[94] Sambariya, D.K., Fagna, R.: A novel elephant herding optimization based PID controller design for load frequency control in power system. In: International Conference on Computer, Communications and Electronics (Comptelix). pp. 595–600 (July 2017)

[95] Schubert, E., Zimek, A.: ELKI data mining (2008), `https://elki-project.github.io/datasets/`

[96] Seker, S.E., Mert, C., Al-Naami, K., Ayan, U., Ozalp, N.: Ensemble classification over stock market time series and economy news. In: IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 272–273. IEEE (2013)

[97] Shao, D., Xu, C., Xiang, Y., Gui, P., Zhu, X., Zhang, C., Yu, Z.: Ultrasound image segmentation with multilevel threshold based on differential search algorithm. IET Image Processing 13(6), 998–1005 (2019)

[98] Shi, Y.: Brain storm optimization algorithm. In: Advances in Swarm Intelligence, LNCS. vol. 6728, pp. 303–309. Springer Berlin Heidelberg (2011)

[99] Sonmez, A., Kocyigit, E., Kugu, E.: Optimal path planning for UAVs using genetic algorithm. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 50–55 (June 2015)

[100] Takahama, T., Sakai, S., Iwane, N.: Constrained optimization by the constrained hybrid algorithm of particle swarm optimization and genetic algorithm. Advances in Artificial Intelligence, LNCS 3809, 389–400 (2005)

[101] Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009), p. 625

[102] Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Advances in Swarm Intelligence, Part I. pp. 355–364. Springer, Berlin Heidelberg (2010)

[103] Ticknor, J.L.: A Bayesian regularized artificial neural network for stock market forecasting. Expert Systems with Applications 40(14), 5501–5506 (2013)

[104] Tuba, E., Mrkela, L., Tuba, M.: Support vector machine parameter tuning using firefly algorithm. In: 2016 26th International Conference Radioelektronika. pp. 413–418 (April 2016)

[105] Tuba, E., Tuba, M., Simian, D.: Adjusted bat algorithm for tuning of support vector machine parameters. In: IEEE Congress on Evolutionary Computation (CEC). pp. 2225–2232 (July 2016)

[106] Tuba, E., Alihodzic, A., Tuba, M.: Multilevel image thresholding using elephant herding optimization algorithm. In: 14th International Conference on Engineering of Modern Electric Systems (EMES). pp. 240–243 (June 2017)

[107] Tuba, E., Tuba, M., Beko, M.: Support vector machine parameters optimization by enhanced fireworks algorithm. In: International Conference in Swarm Intelligence, LNCS. vol. 9712, pp. 526–534. Springer (2016)

[108] Tuba, E., Tuba, M., Beko, M.: Support vector machine parameters optimization by enhanced fireworks algorithm. In: International Conference in Swarm Intelligence, pp. 526–534. Springer (2016)

[109] Tuba, E., Tuba, M., Beko, M.: Support vector machine parameters optimization by enhanced fireworks algorithm. In: Tan, Y., Shi, Y., Niu, B. (eds.) Advances in Swarm Intelligence: 7th International Conference, ICSI, Proceedings, Part I. pp. 526–534. Springer International Publishing, Cham (2016)

[110] Tuba, E., Tuba, M., Dolicanin, E.: Adjusted fireworks algorithm applied to retinal image registration. Studies in Informatics and Control 26(1), 33–42 (2017)

[111] Tuba, E., Tuba, M., Jovanovic, R.: An algorithm for automated segmentation for bleeding detection in endoscopic images. In: International Joint Conference on Neural Networks (IJCNN). pp. 4579–4586. IEEE (2017)

[112] Tuba, M., Alihodzic, A., Bacanin, N.: Cuckoo search and bat algorithm applied to training feed-forward neural networks. In: Yang, X.S. (ed.) Recent Advances in Swarm Intelligence and Evolutionary Computation, Studies in Computational Intelligence, vol. 585, pp. 139–162. Springer International Publishing (2015)

[113] Tuba, M., Bacanin, N., Alihodzic, A.: Multilevel image thresholding by fireworks algorithm. In: 25th International Conference Radioelektronika. pp. 326–330 (April 2015)

[114] Tuba, M., Bacanin, N., Beko, M.: Fireworks algorithm for RFID network planning problem. In: 25th International Conference Radioelektronika. pp. 440–444 (April 2015)

[115] Tuba, M., Jovanovic, R.: Improved ACO algorithm with pheromone correction strategy for the traveling salesman problem. International Journal of Computers, Communications & Control 8(3), 477–485 (June 2013)

[116] Tuba, V., Beko, M., Tuba, M.: Performance of elephant herding optimization algorithm on CEC 2013 real parameter single objective optimization. WSEAS Transactions on Systems 16(13), 100–105 (2017)

[117] Übeyli, E.D., Doğdu, E.: Automatic detection of erythemato-squamous diseases using k-means clustering. Journal of medical systems 34(2), 179–184 (2010)

[118] Vanjak, Z.: Environment for solving of optimization problems. Ph.D. thesis, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu (2006)

[119] Venkatesan, A., Parthiban, L.: Medical image segmentation with fuzzy c-means and kernelized fuzzy c-means hybridized on pso and qpso. International Arab Journal of Information Technology (IAJIT) 14(1) (2017)

[120] Wan, H.: Applying the genetic algorithm to optimization problems. WIT Transactions on Information and Communication Technologies 2, 1–12 (1970)

[121] Wang, G.G., Deb, S., Gao, X.Z., Coelho, L.D.S.: A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. International Journal of Bioinspired Computation 8(6), 394–409 (2016)

[122] Wang, G.G., Deb, S., Gao, X.Z., Coelho, L.D.S.: A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. International Journal of Bio-Inspired Computation 8(6), 394–409 (Jan 2017)

[123] Wang, G., Guo, L., Duan, H., Liu, L., Wang, H.: A bat algorithm with mutation for UCAV path planning. The Scientific World Journal 2012(Article ID 418946), 15 (July 2012)

[124] Wang, G., Guo, L., Duan, H., Liu, L., Wang, H.: A bat algorithm with mutation for ucav path planning. The Scientific World Journal 2012 (2012)

[125] Wang, G., Guo, L., Duan, H., Liu, L., Wang, H.: A modified firefly algorithm for UCAV path planning. International Journal of Hybrid Information Technology 5(3), 123–144 (July 2012)

[126] Wang, G., Guo, L., Duan, H., Liu, L., Wang, H., et al.: A modified firefly algorithm for ucav path planning. International Journal of Hybrid Information Technology 5(3), 123–144 (2012)

[127] Wang, J., Hou, R., Wang, C., Shen, L.: Improved v-support vector regression model based on variable selection and brain storm optimization for stock price forecasting. Applied Soft Computing 49, 164–178 (2016)

[128] Wang, R., Li, Y., Lin, Y., Luo, X., Zhang, X.: Minimum weight triangulation based on improved genetic quantum algorithm. Journal of Computational Information Systems 1(2), 323–327 (2005)

[129] Wang, Y., Cao, Y.: Leukocyte nucleus segmentation method based on enhancing the saliency of saturation component. Journal of Algorithms & Computational Technology 13, 1748302619845783 (2019)

[130] Wu, K.P., Wang, S.D.: Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. Pattern Recognition 42(5), 710–717 (2009)

[131] Wu, Q.: A self-adaptive embedded chaotic particle swarm optimization for parameters selection of wv-svm. Expert Systems with Applications 38(1), 184–192 (2011)

[132] Xie, J., Xie, W., Wang, C., Gao, X.: A novel hybrid feature selection method based on IFSFFS and SVM for the diagnosis of erythemato-squamous diseases. In: Proceedings of the First Workshop on Applications of Pattern Analysis. pp. 142–151 (2010)

[133] Xu, C., Duan, H., Liu, F.: Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning. Aerospace Science and Technology 14(8), 535–541 (2010)

[134] Yang, X.S.: Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. pp. 169–178. Springer (2009)

[135] Yang, X.S.: Engineering optimization: an introduction with metaheuristic applications. John Wiley & Sons (2010)

[136] Yang, X.S.: A new metaheuristic bat-inspired algorithm. Studies in Computational Intelligence 284, 65–74 (November 2010)

[137] Yang, X.S.: Cuckoo search and firefly algorithm: theory and applications, vol. 516. Springer (2013)

[138] Yang, X.S., Deb, S.: Cuckoo search via Levy flights. In: World Congress on Nature Biologically Inspired Computing, 2009. NaBIC 2009. pp. 210–214 (Dec 2009)

[139] Yang, X.S., Hossein Gandomi, A.: Bat algorithm: a novel approach for global engineering optimization. Engineering Computations 29(5), 464–483 (2012)

[140] Yang, X., Bramer, M., Ellis, R., Petridis, M.: Research and development in intelligent systems xxvi. Incorporating Applications and Innovations in Intelligent Systems XVII, Springer London pp. 209–218 (2010)

[141] Yousefi, A., Young, N.E.: On a linear program for minimum-weight triangulation. SIAM Journal on Computing 43(1), 25–51 (2014)

[142] Zamuda, A., Sosa, J.D.H.: Differential evolution and underwater glider path planning applied to the short-term opportunistic sampling of dynamic mesoscale ocean structures. Applied Soft Computing 24, 95–108 (2014)

[143] Zhang, B., Duan, H.: Three-dimensional path planning for uninhabited combat aerial vehicle based on predator-prey pigeon-inspired optimization in dynamic environment. IEEE/ACM transactions on computational biology and bioinformatics 14(1), 97–107 (2015)

[144] Zhang, H., Berg, A.C., Maire, M., Malik, J.: SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 2, pp. 2126–2136 (2006)

[145] Zhang, Y., Wang, S., Phillips, P., Ji, G.: Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. Knowledge-Based Systems 64, 22–31 (2014)

[146] Zheng, S., Janecek, A., Tan, Y.: Enhanced fireworks algorithm. In: IEEE Congress on Evolutionary Computation (CEC). pp. 2069–2077 (June 2013)

[147] Zheng, S., Li, J., Janecek, A., Tan, Y.: A cooperative framework for fireworks algorithm. IEEE/ACM Transactions on Computational Biology and Bioinformatics 14(1), 27–41 (2017)

[148] Zhong, N., Liu, J., Yao, Y.Y., Ohsuga, S.: Web intelligence (wi). In: The 24th Annual International Conference on Computer Software and Applications (COMPSAC). pp. 469–470. IEEE (2000)

[149] Zhu, W., Duan, H.: Chaotic predator–prey biogeography-based optimization approach for UCAV path planning. Aerospace Science and Technology 32(1), 153–161 (2014)

PRILAGOĐAVANJE ALGORITAMA INTELIGENCIJE ROJEVA ZA RAZLIČITE PROSTORE PRETRAGE

ORIGINALITY REPORT

6%

SIMILARITY INDEX

PRIMARY SOURCES

1	nardus.mpn.gov.rs Internet	957 words — 2%
2	"Harmony Search and Nature Inspired Optimization Algorithms", Springer Science and Business Media LLC, 2019 Crossref	748 words — 2%
3	Eva Tuba, Raka Jovanovic, Romana Capor Hrosik, Adis Alihodzic, Milan Tuba. "Web Intelligence Data Clustering by Bare Bone Fireworks Algorithm Combined with K-Means", Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics - WIMS '18, 2018 Crossref	112 words — < 1%
4	Eva Tuba, Ivana Ribic, Romana Capor-Hrosik, Milan Tuba. "Support Vector Machine Optimized by Elephant Herding Algorithm for Erythemato-Squamous Diseases Detection", Procedia Computer Science, 2017 Crossref	105 words — < 1%
5	repozitorij.unios.hr Internet	63 words — < 1%
6	link.springer.com Internet	44 words — < 1%
7	repozitorij.unizg.hr Internet	36 words — < 1%

8	Ying Tan. "chapter 1 Recent Developments of Fireworks Algorithms", IGI Global, 2020 Crossref	32 words — < 1%
9	www.bib.irb.hr Internet	31 words — < 1%
10	"A Modified Elephant Herd Optimization Algorithm to Solve the Single Machine Scheduling Problems", International Journal of Engineering and Advanced Technology, 2019 Crossref	30 words — < 1%
11	Ferdinando Di Martino, Salvatore Sessa. "Chapter 6 Fuzzy Transform for Image Segmentation", Springer Science and Business Media LLC, 2020 Crossref	30 words — < 1%
12	transpireonline.blog Internet	30 words — < 1%
13	Adis Alihodzic, Eva Tuba, Romana Capor-Hrosik, Edin Dolicanin, Milan Tuba. "Unmanned aerial vehicle path planning problem by adjusted elephant herding optimization", 2017 25th Telecommunication Forum (TELFOR), 2017 Crossref	26 words — < 1%
14	Islam A. ElShaarawy, Essam H. Houssein, Fatma Helmy Ismail, Aboul Ella Hassanien. "An exploration-enhanced elephant herding optimization", Engineering Computations, 2019 Crossref	26 words — < 1%
15	ascelibrary.org Internet	21 words — < 1%
16	epdf.tips Internet	20 words — < 1%
17	"Advances in Harmony Search, Soft Computing and Applications", Springer Science and Business	11 words — < 1%

Media LLC, 2020
Crossref

18 "Advances in Swarm Intelligence", Springer Science and Business Media LLC, 2017 — 10 words — < 1%
Crossref

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	< 15 WORDS
EXCLUDE BIBLIOGRAPHY	ON		