Pregled i analiza najnovijih prirodom inspiriranih metaheurističkih algoritama u računarskoj znanosti

Kvaternik, Vilim

SVEUČILIŠTE U RIJECI

POMORSKI FAKULTET U RIJECI

VILIM KVATERENIK

PREGLED I ANALIZA NAJNOVIJIH PRIRODOM INSPIRIRANIH METAHEURISTIČKIH ALGORITAMA U RAČUNARSKOJ ZNANOSTI

DIPLOMSKI RAD

Rijeka, 2022.

SVEUČILIŠTE U RIJECI

POMORSKI FAKULTET U RIJECI

PREGLED I ANALIZA NAJNOVIJIH PRIRODOM INSPIRIRANIH METAHEURISTIČKIH ALGORITAMA U RAČUNARSKOJ ZNANOSTI

AN OVERVIEW AND ANALYSIS OF THE LATEST NATURE-INSPIRED METAHEURISTIC ALGORITHMS IN COMPUTER SCIENCE

DIPLOMSKI RAD

Kolegij: Algoritmi i strukture podataka

Mentor: Doc.dr.sc. Marko Gulić

Student: Vilim Kvaternik

Studijski smjer: Elektroničke i informatičke tehnologije u Pomorstvu

JMBAG: 0112069800

Rijeka, rujan 2022.

Student: Vilim Kvaternik

Studijski program: Elektroničke i informatičke tehnologije u Pomorstvu

JMBAG: 011206800

IZJAVA O SAMOSTALNOJ IZRADI DIPLOMSKOG RADA

Kojom izjavljujem da sam diplomski rad s naslovom

PREGLED I ANALIZA NAJNOVIJIH PRIRODOM INSPIRANIH METAHEURISTIČKIH ALGORITAMA U RAČUNARSKOJ ZNANOSTI

(naslov diplomskog rada)

izradio/la samostalno pod mentorstvom

Doc. dr. sc. Marko Gulić

(prof. dr. sc. / izv. prof. dr. sc. / doc dr. sc Ime i Prezime)

te komentorstvom _____

stručnjaka/stručnjakinje iz tvrtke _____
(naziv tvrtke).

U radu sam primijenio/la metodologiju izrade stručnog/znanstvenog rada i koristio/la literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo/la u diplomskom radu na uobičajen, standardan način citirao/la sam i povezao/la s fusnotama i korištenim bibliografskim jedinicama, te nijedan dio rada ne krši bilo čija autorska prava. Rad je pisan u duhu hrvatskoga jezika.

Student/studentica

_____

(potpis)

Ime i prezime studenta/studentice

Student: Vilim Kvaternik

Studijski program: Elektroničke i informatičke tehnologije u Pomorstvu

JMBAG: 011206800

IZJAVA STUDENTA – AUTORA
O JAVNOJ OBJAVI OBRANJENOG DIPLOMSKOG RADA

Izjavljujem da kao student – autor diplomskog rada dozvoljavam Pomorskom fakultetu Sveučilišta u Rijeci da ga trajno javno objavi i besplatno učini dostupnim javnosti u cjelovitom tekstu u mrežnom digitalnom repozitoriju Pomorskog fakulteta.

U svrhu podržavanja otvorenog pristupa diplomskim radovima trajno objavljenim u javno dostupnom digitalnom repozitoriju Pomorskog fakulteta, ovom izjavom dajem neisključivo imovinsko pravo iskorištavanja bez sadržajnog, vremenskog i prostornog ograničenja mog diplomskog rada kao autorskog djela pod uvjetima *Creative Commons* licencije CC BY Imenovanje, prema opisu dostupnom na http://creativecommons.org/licenses/

Student/studentica - autor

(potpis)

SAŽETAK

U ovom diplomskom radu napravljen je pregled i analiza najnovijih prirodom inspiriranih metaheurističkih algoritama u računalnoj znanosti. U prvom dijelu rada predstavljena je metaheuristička znanost od njenih početaka pa sve do današnjih spoznaja. Nadalje, u drugom dijelu rada općenito su predstavljeni metaheuristički algoritmi te njihova klasifikacija. Posljednji dio rada prikazuje i analizira šest najnovijih metaheurističkih algoritama inspiriranih prirodom: algoritam za optimizaciju lova na jelene, metaheuristički algoritam baziran na plavog majmuna, metaheuristički algoritam pretrage temeljen na medvjeđem njuhu, metaheuristički algoritam inspiriran jelenima, optimizacija temeljena na surom orlu, optimizacijski algoritam pretraživanja temeljen na gušteru.

Ključne riječi: metaheuristika, metaheuristički algoritmi inspirirani prirodom, optimizacija.

SUMMARY

In this thesis, an overview and analysis of the latest nature-inspired metaheuristic algorithms in computer science was made. The first part of the paper presents metaheuristic science from its beginnings to the present day. Furthermore, in the second part of the paper metaheuristic algorithms and their classification are generally presented. The last part of the paper presents and analyses six latest metaheuristic algorithms inspired by nature: deer hunting optimization algorithm, blue monkey algorithm, bear smell search algorithm, red deer algorithm, golden eagle optimizer, artificial lizard search optimization

Keywords: metaheuristics, metaheuristic algorithms inspired by nature, optimization.

SADRŽAJ

1. UVOD

Optimizacija igra ključnu ulogu u svakodnevnom životu i to je preneseno na različite aplikacije u inženjeringu, industrijskom dizajnu i poslovanju, gdje ciljevi mogu biti bilo što, poput minimiziranja potrošnje energije i troškova, ili maksimiziranja dobiti, učinkovitosti i produktivnosti. Budući da su novac i vrijeme uvijek nedovoljni, potrebno je pronaći rješenja za optimalno korištenje tih vrijednih resursa. Kako bi riješili brojna ograničenja u aplikacijama, matematički alati su se pokazali kao dobra opcija za rješavanje tih problema dizajna.

Jedna od osnovnih značajka živih bića u prirodi je društveno ponašanje koje pomaže u obavljanju različitih zadataka. Opstanak je krajnji cilj svakog pojedinačnog i kolektivnog ponašanja. Životinje tvore skupine, čopore, stada, jata i sl., u različite svrhe poput lova, migracije, obrane itd. Moguće je razviti algoritme koji su nadahnuti načinom na koji životinje pronalaze najbolja rješenja za probleme optimizacije.

Primijećeno je da klasične metode nisu u stanju pružiti željeno riješenje optimizacije s dovoljnom točnošću i u razumnom roku. Posljednjih nekoliko desetljeća došlo je do značajnog porasta broja algoritama optimizacije inspiriranih prirodom, oponašajući različite fizičke pojave i biološko ponašanje životinja.

Glavni razlog za razvoj novih metaheuristika je taj što ne postoji metaheuristički algoritam optimizacije za rješavanje svih problema optimizacije. To znači da uvijek postoji mogućnost da novi metaheuristički algoritam pokaže bolje performanse u odnosu na neke druge algoritme za određeni optimizacijski problem.

U ovom radu bit će predstavljeno šest najnovijih prirodom inspiriranih metaheurističkih algoritama u računalnoj znanosti. Prvi je algoritam za optimizaciju lova na jelene gdje se lovac pokušava što bolje pozicionirati u prostoru za lov na jelene. Drugi je algoritam temeljen na plavom majmunu gdje se određuje koliko ima mužjaka u jednoj skupini plavih majmuna. Treći je algoritam je temeljen na medvjeđem njuhu gdje se kroz olfaktivne organe pokušava predvidjeti kvaliteta mirisa iz skupa komponenata mirisa. Četvrti je algoritam inspiriran jelenima koji predstavlja neobično ponašanje parenja škotskog Jelena. Peti je optimizacija temeljena na surom orlu koji oponaša faze leta i napada (lova) surog orla. Šesti je algoritam temeljen na gušteru gdje se pokušava modelirati ponašanje *Agame* guštera u potrazi za hranom i njihov učinkovit način hvatanja plijena.

2. POVIJEST METAHEURISTIKE

Iz starogrčkog di metaheuristika znaci „meta“ („visa razina“) i „heurískō“ („pronađem, otkrijem“). Područje (meta) heuristike, posebno u usporedbi s drugim područjima istraživanja poput fizike, kemije i matematike, još nije dostigla svoj cijeli potencijal [1]. Ljudi su koristili heuristiku i metaheuristiku mnogo prije nego što je taj pojam nastao. Sam pojam uveden je u drugoj polovici 1980.-ih Način na koji su ljudi (ne samo istraživači) tumačili različite metaheurističke koncepte odredilo je smjer u kojem se ova grana znanosti razvija.

Zbog pojednostavljivanja, povijest metaheuristike se dijeli na pet različitih razdoblja. Stvarna vremenska razdoblja tijekom kojih su se dogodile promjene paradigme obično su udaljena nekoliko godina.

- **Pred-teorijsko** razdoblje (prije 1940.) tijekom kojeg se heuristika, pa čak i metaheuristika koriste, ali nisu formalno proučavane.
- **Rano razdoblje** (oko 1940. – oko 1980.), tijekom kojeg se pojavljuju prve formalne studije o heuristici.
- **Razdoblje orijentirano na metode** (od 1980. - do 2000.) tijekom kojeg se polje metaheuristike istinski razvija i predlaže se mnogo različitih metoda.
- **Razdoblje orijentirano na razvojno okruženje (razvojni okvir)** (od 2000. – do danas), tijekom kojeg raste svijest da je metaheuristiku korisnije opisivati kao razvojni okvir, a ne kao metodu.
- **Znanstveno razdoblje** (budućnost) tijekom kojeg razvoj metaheuristike postaje znanost,

2.1.1 Pred-teorijsko razdoblje

Optimizacijski problem okružuju nas oduvijek sa svih strana. Kontinuirano se odlučuje, što se najviše isplati pri kupovini, kojim putem će se najbrže stići do nekog cilja izbjegavajući gužvu, u što treba investirati da bi se došlo do profita, prepoznavanje ljudi iz daljine, itd. U osnovi, riješavaju se problemi optimizacije. Za ljude (i mnoge životinjske vrste) rješavanje problema optimizacije ne zahtijeva nikakvu formalnu obuku. Sposobnost adekvatnog i brzog rješavanja problema jedan je od najvažnijih čimbenika koji određuje vjerojatnost preživljavanja. Ljudski (i životinjski) um rješava probleme optimizacije heuristički, a ne precizno, tj. rješenja koja proizvodi mozak nisu obavezno optimalna.

Razlika između točnih rješenja i približnih rješenja, razlika između jednostavnih i složenih problema optimizacije ili između brzih (polinomnih) i sporih (eksponencijalnih) algoritama diskutabilna je za prosječnu osobu. Jedini način na kojem se može saznati je li ljudska optimizacija točna je provjeriti rezultat tog događanja i usporediti ga sa ranije napravljenom optimizacijom.

Moglo bi se pretpostaviti da je čovjeku heuristika toliko prirodna da je trebalo pričekati dok se ne razvije formalna teorija optimizacije, posebno linearnog programiranja, prije nego što ju je itko smatrao vrijednom proučavanja.

2.1.2 Rano razdoblje

U 1945., neposredno nakon Drugog svjetskog rata, mađarski matematičar George Pólya u svojoj knjizi *„How to solve it“* tvrdi da se problemi mogu riješiti ograničenim skupom općenito primjenjivih strategija, koje služe za olakšavanje rješavanja problema. Koliko god je njegova strategija rješenja bila namijenjena za rješavanje matematičkih problema, jednako je primjenjiva i na razvoj algoritama za optimizaciju.

Tako načelo "analogije", na primjer, govori rješavaču problema da traži drugi problem koji je vrlo sličan problemu koji se razmatra i za koji je poznata metoda rješavanja. Proučavanjem sličnosti i razlika između oba problema mogu se dobiti ideje za rješavanje izvornog problema. Princip "indukcije" za rješavanje problema izvođenjem generalizacije iz nekih primjera. Ideja "pomoćnog problema" traži pomoćni problem koji može pomoći u rješavanju općeg problema. Iz ovoga se može vidjeti kako su ova načela bila korisna za razvoj metaheurističkih algoritama.

Ono što je važno znati je da niti jedna strategija Pólya zapravo ne rješava nijedan problem, niti se same po sebi mogu nazvati "algoritmima". Umjesto toga, oni su metastrategije na visokoj razini koje su korisne za utjecaj na to kako učenik heuristike razmišlja o problemu.

U tom razdoblju pojavilo se i nekoliko algoritamskih ideja vrlo visoke razine. Konstruktivni algoritam je algoritam koji započinje praznim rješenjem i iterativno dodaje jedan po jedan element dok se ne stvori cjelovito rješenje. Pohlepno pravilo odabira, odabire najbolju stavku na svakoj iteraciji [2].

Iako je heuristika razvijena u ranom razdoblju bila vrlo jednostavna, spoznaja da postoje strategije na visokoj razini koje se mogu koristiti kao osnova za razvoj heuristike (problem optimizacije), dovela je do spoznaja koje su otvorile put složenijim metastrategijama.

2.1.3 Razdoblje orijentirano na metode

Počevši od 1960.-ih pojavio se potpuno drugačiji način istraživanja metoda rješavanja problema. Te metode koriste „evoluciju“ kako bi moglo rješavati životne probleme. Ipak, trebalo je još jedno stoljeće i pojava računala prije nego što su se istraživači zainteresirali za oponašanje procesa prirodne evolucije [3].

Krajem 1950.-ih i početkom 1960.-ih razvijeno je ono što bismo danas nazvali evolucijskim algoritmima. Njihov glavni cilj nije bio rješavanje problema optimizacije, već proučavanje fenomena prirodne evolucije. Evolucijsko programiranje uvedeno nekoliko godina kasnije, predstavljalo je rješenja u obliku konačnih strojeva [4]

Pravi početak područja evolucijskih algoritama postavio je seminarski rad Johna Hollanda. Sa svojom shemom-teoremom koja u osnovi tvrdi da će rješenja postati kvalitetnija sa povećanjem frekvencije uzastopnih iteracija algoritma [5].

Vrlo važno je bilo pronalaženje generičke metode heurističke optimizacije koja je mogla učinkovito riješiti problem bez potrebe za informacijama o određenom problemu. 1980.-ih, počinju se pojavljivati prvi članci koji uvode generička okruženja za rješavanje problema koji se ne temelje na prirodnoj evoluciji. Jedan od prvih je bio Kirkpatrick. On je pomoću simuliranog kaljenja, koji se koristi u metalurgiji i proizvodnji stakla za ublažavanje stresa materijala, simulirao kaljenje koristeći nasumične promjene u otopinama [6].

Možda je najutjecajnija metoda koja se temelji na umjetnoj inteligenciji bila Tabu pretraga [7]. Osnovna pretpostavka ovog razvojnog okvira je da se lokalni algoritam pretraživanja može usmjeriti prema dobrom rješenju koristeći neke informacije prikupljene tijekom pretraživanja u prošlosti. U tu svrhu, tabu pretraga identificirala je niz memorijskih struktura koje bilježe aspekte pretraživanja. Najsimboličniji je tabu lista, popis koji bilježi atribute rješenja i zabranjuje sva rješenja koja imaju atribut na tabu listi za određeni broj ponavljanja. U ovom radu također je uvedena riječ "metaheuristički", iako još nije bila korištena na način kao što je danas.

Zanimljivo je da su neuronske mreže bile među ograničenim popisom metaheuristika predloženih krajem 1980.-ih. Ove tehnike oponašaju funkcioniranje mozga (uključujući

neurone i sinapse) i izvorno su predložene u kontekstu prepoznavanja uzoraka (za što se još uvijek uglavnom koriste) [8].

U drugoj polovici 1990.-ih započelo je sustavno pažljivo proučavanje performansi i ponašanja heuristike, poput evolucijskih algoritama i optimizacije kolonija mrava. Te su studije identificirale i jednostavne zadatke u kojima heuristika dobro funkcionira i jednako jednostavne zadatke u kojima ne funkcionira te se zahtijeva eksponencijalno vrijeme optimizacije. Osim toga, oni bi mogli strogo dokazati kako heuristika može učinkovito optimizirati nekoliko klasičnih kombinatornih problema i kako mogu pružiti dobre aproksimacije za NP teške probleme [9].

Međutim, u istom razdoblju, postupno je postalo jasno da metaheuristika koja se temelji na metaforama neće nužno dovesti do dobrih pristupa. Iako su rani metaheuristički razvojni okviri nudili neke zanimljive uvide, oni nisu eliminirali potrebu za iskusnim heurističkim operaterom. Pojava metaheuristike nije promijenila jednostavnu činjenicu da će metaheuristika koja široko koristi karakteristike razmatranog problema optimizacije gotovo uvijek nadmašiti onu koja koristi pristup "crne kutije" (procjenjuje sustav isključivo izvana, a da operater ili ispitivač ne znaju što se događa unutar sustava), bez obzira na metaheuristički razvojni okvir koji se koristi.

Općenito, istraživači su u razdoblju, koji je orijentiran na metode predložili „algoritme“, odnosno formalizirane strukture koje je trebalo slijediti, poput recepta za kuharicu.

2.1.4 Okvirno orijentirano razdoblje

Dok su se istraživači prijašnjih razdoblja ograničavali na jednu metaheurističku strukturu, na prijelazu stoljeća sve je više istraživača kombiniralo ideje iz različitih struktura u jedan heuristički algoritam.

Jedna vrsta hibridne metaheuristike čak je dobila i zasebno ime: upotreba lokalnog pretraživanja (ili bilo kojeg pristupa „lokalnom učenju“) za poboljšanje rješenja dobivenih evolucijskim algoritmom nazvana je „memetičkim“ algoritmom [10].

Otkriće pojedinačnih algoritamskih okvira omogućilo je istraživačima kombiniranje metaheuristike s bilo kojom dostupnom pomoćnom metodom. Ograničeno programiranje, linearno programiranje i *mixed-integer* programiranje korišteni su zajedno s idejama metaheuristike. Koncept "metaheurističkih struktura" implicirao je da metaheuristika nije ništa

drugo nego koherentan skup ideja koje se, naravno, mogu slobodno kombinirati s drugim idejama. Danas mnogi istraživači razvijaju metaheuristiku koristeći svoje iskustvo i znanje o tome koje će tehnike dobro funkcionirati za rješavanje određenih problema, a koje najvjerojatnije neće.

Kombinacijom najučinkovitijih operatora postojećih metaheurističkih struktura i pažljivim podešavanjem rezultirajuće heuristike, mogu se stvoriti algoritmi koji učinkovito rješavaju bilo koji stvarni problem optimizacije.

Iako je znanstvena zajednica postigla značajan napredak u svojoj potrazi za razumijevanjem temeljnog ponašanja metaheuristike, metaheuristička zajednica tradicionalno stavlja veliki naglasak na izvedbu. Studija se smatra dobrom ako (i samo ako) daje heuristički algoritam koji se dobro "izvodi" u odnosu na neku referentnu vrijednost, kako što je druga heuristika ili donja granica.

2.1.5 Znanstveno razdoblje

Kroz razdoblja, grana znanosti metaheuristike nije se shvaćala ozbiljno. Tradicionalno, teorijska osnova heuristike i metaheuristike nije bila u rangu s drugim teorijama ili, točnije, sa stvarnim metodama. Razvoj algoritama za heurističku optimizaciju, bez obzira koristi li metaheurističku strukturu ili ne, temelji se na iskustvu, a ne na teoriji.

Rani pokušaji da se u teoriji čvrsto potkrepi razvoj metaheuristike nisu ispunili svoja obećanja. Razumijevanje ponašanja metaheuristike na temeljnoj razini pokazalo se teškim zadatkom, unatoč nekoliko zapaženih napora [11].

Najvažnije je prijelaz iz zajednice orijentiranu na izvedbu metaheuristike u znanstvenog razumijevanja, dogodit će se tijekom znanstvenog razdoblja. Bez sumnje, to će dovesti do razvoja boljih i učinkovitijih metaheuristika, ali će također dovesti do metaheuristike koja se može koristiti izvan laboratorija.

3. METAHEURISTIČKI ALGORITMI

Metaheuristički algoritmi bave se problemima optimizacije. Optimizacija se događa gotovo svugdje, od inženjerskog dizajna do ekonomije, od planiranja odmora do internetskog usmjeravanja. Budući da su novac, resursi i vrijeme uvijek ograničeni, optimalno korištenje tih raspoloživih resursa je presudno [12].

Nažalost, većina problema optimizacije klasificirana je kao NP-teški problemi koji se ne mogu riješiti u polinomijalnom vremenu sve dok je NP jednako P (P uključuje sve probleme koji se mogu učinkovito riješiti, a NP uključuje sve probleme koji se, koji su nerješivi, ali može se učinkovito provjeriti je li rješenje ispravno). Na taj se način mogu obraditi samo primjerci s malom skalom preciznih matematičkih metoda. Umjesto da odustanu, istraživači su odlučili iskoristiti moguće zaobilazna rješenja (metode aproksimacije) koja u razumnom vremenu mogu pronaći dovoljno dobro rješenje. Ta se zaobilazna rješenja mogu podijeliti na heuristička i metaheuristička.

Heuristička ili metaheuristička metoda je svaki pristup rješavanju problema ili samospoznaji koji koristi praktičnu metodu za koju nije zajamčeno da je optimalna, savršena ili racionalna, ali je ipak dovoljna za postizanje neposrednog, kratkoročnog cilja ili aproksimacije.

Većina konvencionalnih i klasičnih algoritama su deterministički. Na primjer, jednostavna metoda u linearnom programiranju je deterministička. Neki algoritmi determinističke optimizacije koristili su informacije o gradijentu; oni se nazivaju algoritmi temeljeni na gradijentu. Na primjer, dobro poznati Newton-Raphson [13] algoritam temelji se na gradijentu, jer koristi vrijednosti funkcija i njihove derivate, a izuzetno dobro funkcionira za glatke unimodalne probleme. Međutim, ako postoji određena rupa u ciljnoj funkciji, taj algoritam loše funkcionira. U ovom je slučaju poželjan algoritam bez gradijenta. Algoritmi koji ne koriste gradijente ne koriste nikakve derivate, već samo vrijednosti funkcija.

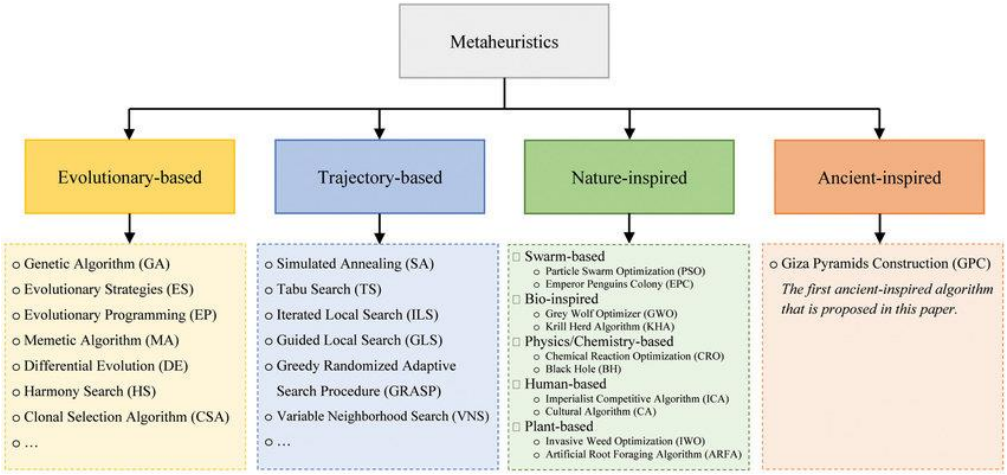
Generalno stohastičke algoritme dijelimo na dvije vrste: heurističke i metaheurističke, iako je njihova razlika mala. Grubo govoreći, heuristika znači "pronaći" ili "otkriti pokušajem i pogreškom". Kvalitativna rješenja složenog problema optimizacije mogu se naći u razumnom vremenskom razdoblju, ali ne postoji jamstvo da će se postići optimalna rješenja. Očekuje se da ovi algoritmi rade većinu vremena, ali ne cijelo vrijeme. To je dobra stvar kada nam ne trebaju nužno bolja rješenja, već dobra rješenja koja su lako dostupna.

Daljnji razvoj heurističkih algoritama su takozvani metaheuristički algoritmi. Ovdje meta-znači "izvan" ili "viša razina", a ti algoritmi obično rade bolje od jednostavnih heuristika. Nadalje, svi metaheuristički algoritmi koriste određeni kompromis između randomizacije i lokalnog pretraživanja. Vrijedno je napomenuti da u literaturi ne postoje dogovorene definicije heuristike i metaheuristike; neki koriste heuristiku i metaheuristiku naizmjenično. Međutim, u posljednje vrijeme postoji tendencija da se svi stohastički algoritmi s randomizacijom i lokalnim pretraživanjem nazivaju metaheurističkim. Randomizacija je dobar način za prelazak s lokalnog pretraživanja na globalno pretraživanje. Gotovo svi metaheuristički algoritmi dizajnirani su za globalnu optimizaciju.

Heuristika je način pokušaja i pogrešaka za pronalaženje prihvatljivih rješenja složenog problema u razumno kratkom vremenu. Složenost problema otežava mogućnost pronalaženja mogućih rješenja ili kombinacija. Cilj je pronaći dobro izvedivo rješenje u prihvatljivom vremenskom okviru. Ne postoji jamstvo da će se naći najbolje rješenje, a niti u procesu stvaranja ne zna se hoće li određeni algoritam raditi. Ideja je imati učinkovit, ali praktičan algoritam koji će raditi većinu vremena i koji je u stanju stvoriti kvalitetna rješenja. Među pronađenim kvalitetnim rješenjima očekuje se da će neka od njih biti gotovo optimalna, iako ne postoji jamstvo takve optimalnosti.

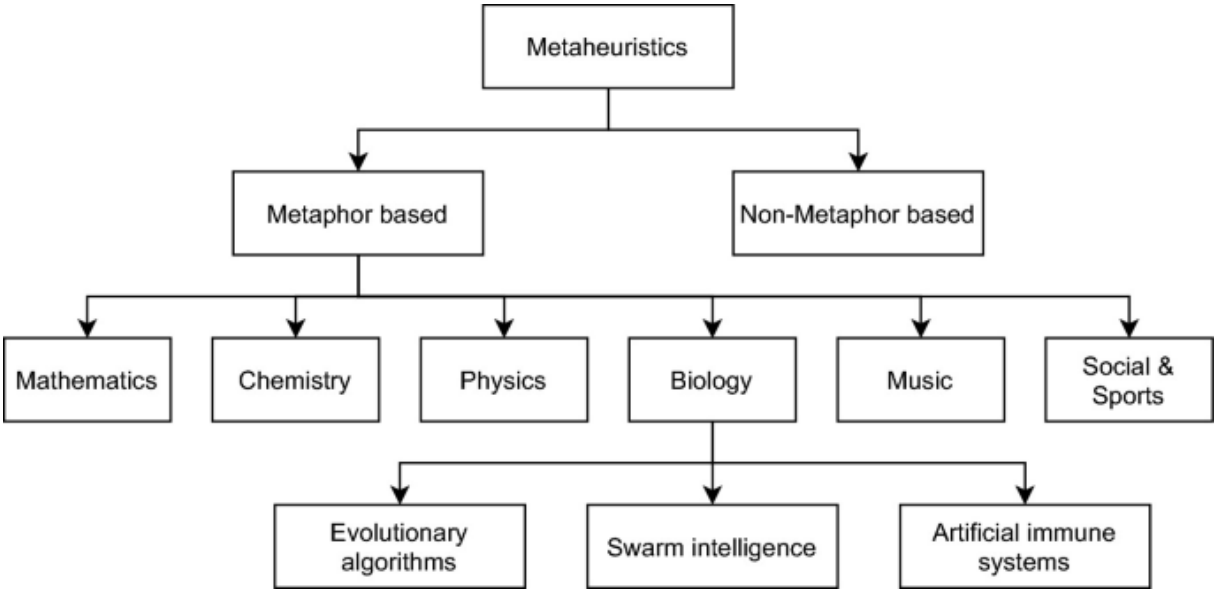
Dvije glavne komponente bilo kojeg metaheurističkog algoritma su eksploatacija i istraživanje. Istraživanje podrazumijeva stvaranje različitih rješenja za istraživanje prostora pretraživanja na globalnoj razini. Eksploatacija znači usredotočiti se na pretraživanje u lokalnoj regiji. Koriste se informacije trenutno dobrog rješenja koje je moguće i najbolje rješenje. Odabir najboljeg rješenja osigurava da rješenja konvergiraju prema optimalnosti, dok istraživanje sa randomizacijom izbjegava zarobljavanje rješenja u lokalne optimume i istodobno povećava raznolikost rješenja. Dobra kombinacija ove dvije glavne komponente obično osigurava postizanje globalne optimalnosti.

Metaheuristički algoritmi mogu se klasificirati na različite načine ovisno o literaturi koja se čita kao na Slici 1.



Slika 1. Klasifikacije metaheurisike [14]

Dok klasifikacija koja se koristi u ovom radu dijeli se na metafornu metaheuristiku i ne-metafornu metaheuristiku (Slika 2). U nastavku poglavlja bit će detaljnije objašnjena ova podjela.



Slika 2. Dijeljenje metaheuristike [15]

3.1 METAHEURISTIKA GENERALNO

Grubo rečeno, metaheuristika se smatra algoritamskom strukturom koja se obično primjenjuje na različite probleme optimizacije, s malim izmjenama, kako bi se prilagodili zadanom zadatku. Nadalje, metaheuristika ima temeljne karakteristike koje se mogu sažeti kako slijedi:

- Metaheuristika nije namijenjena rješavanju određenog problema.
- Metaheuristika je obično aproksimacija.
- Metaheuristika istražuje prostor pretraživanja kako bi pronašla "dovoljno dobro" rješenje.
- Metaheuristika se u osnovi može opisati slojem apstrakcije.
- Metaheuristika obično omogućuje jednostavnu istodobnu implementaciju.
- Metaheuristika se proteže od osnovnog lokalnog pretraživanja do naprednih metoda učenja.
- Metaheuristika može uključivati različite mehanizme kako bi se izbjegla preuranjena konvergencija (temeljeno na genetskim algoritmima).
- Heuristika sa metaheuristikom može se koristiti kao znanje koje se odnosi na određenu domenu kojom dominira strategija visoke razine.
- Nove metaheuristike koriste orijentirnu memoriju koja čuva iskustvo pretraživanja.

Kao što je ranije spomenuto, istraživanje i eksploatacija glavne su funkcije metaheuristike, a ključ učinkovitog procesa pretraživanja je pravilan kompromis između istraživanja i eksploatacije.

Kako je prije rečeno, danas se klasifikacija metaheuristike dijeli na metafornu i nemetafornu metaheuristiku. Metaheuristika koja se temelji na metaforama su algoritmi koji modeliraju prirodne pojave, ljudsko ponašanje u modernom stvarnom životu ili čak matematiku i sl. S druge strane, nemetaforna metaheuristika ne koristi nikakvo modeliranje kako bi definirala svoju strategiju pretraživanja.

Metaheuristika se dijeli na:

1) **Metaforna Metaheruristika**:
   a) *Biologija*:
      i) <u>Evolucijski algoritmi:</u>
         Predstavnik ove vrste algoritma: - Genetski algoritam (eng. Genetic Algorithm, GA)
      ii) <u>Inteligencija temeljena na roj</u>:
         Predstavnik ove vrste algoritma: - Optimizacija roja čestica (eng. Particle Swarm Optimization, PSO)
      iii) <u>Umjetni imunološki sustavi</u>:

Predstavnik ove vrste algoritma: - Algoritam klonske selekcije (eng. Clonal Selection Algorithm, CLONALG)

b) *Kemija:*
   i) Optimizacija kemijskih reakcija (eng. Chemical Reaction Optimization, CRO)
   ii) Optimizacija Brownovog kretanja plinova (eng. Gases Brownian Motion Optimization, GBMO)
c) *Glazba:*
   i) Optimizaciski algotiram za pretraživanje harmonije (eng. Harmony search, HS)
   ii) Metoda glazbene kompozicije (eng. Method of Musical Composition, MMC)
d) *Matematika*:
   i) Osnovni algoritam optimizacije (eng. Base Optimization Algorithm, BOA)
   ii) Sinusno-Kosinusni algoritam (eng. Sine Cosine Algorithm, SCA)
e) *Fizika:*
   i) Simulirano kaljenje (eng. Simulated Annealing, SA)
   ii) Algoritam gravitacijskog pretraživanja (eng. Gravitational Search Algorithm, GSA)
f) *Društvena i sportska*:
   i) Optimizacija utemeljena na nastavnom učenju (eng. Teaching–Learning-Based Optimization, TLBO)
   ii) Algoritam ligaškog prvenstva (eng. League Championship Algorithm, LCA)

2) **Ne-metaforna Metaheuristika**:
   a) Tabu pretraga (eng. Tabu search, TS)
   b) Pretraga promjenjivih susjedstva (eng. Variable neighborhood search, VNS)
   c) Djelomična optimizacija metaheuristike pod posebnim intensifikacijskim uvjetma (eng. Partial optimization metaheuristic under special intensification conditions, POPMUSIC)

3.1.1 Metaheuristika zasnovana na metaforama

3.1.1.1 Metaheuristika utemeljena na biologiji

Većina metaheuristike temelji se na načelima biološke evolucije. Konkretno, one su povezani s modeliranjem različitih bioloških metafora koje se razlikuju po načinu predstavljanja (struktura, komponente itd.). Postoje tri glavne paradigme: evolucijski, rojni i imunološki sustav.

Evolucijski algoritmi (eng. Evolutionary algorithm, EA) modeliraju biološku evoluciju na staničnoj razini, koristeći operatore selekcije, križanja, mutacije i reprodukcije kako bi stvorili što bolja primarna rješenja (npr. kromosom). Postoje četiri povijesne paradigme za evolucijsko računanje: evolucijsko programiranje [4], evolucijske strategije [16], genetski algoritmi [17] i genetsko programiranje [18].

Inteligencija rojeva (eng. Swarm inteligence, SI) oponaša kolektivno ponašanje agenata u zajednici, poput ptica i insekata. Inteligencija roja uglavnom ovisi o principu decentralizacije, tj. primarna rješenja ažuriraju se lokalnom interakcijom jedni s drugima i sa svojim okruženjem. Najpopularniji SI algoritmi su optimizacija roja čestica (eng. Particle Swarm Optimization, skraćeno PSO) [19] i optimizacija kolonije mrava (eng. Ant Colony Optimization, skraćeno ACO) [20]

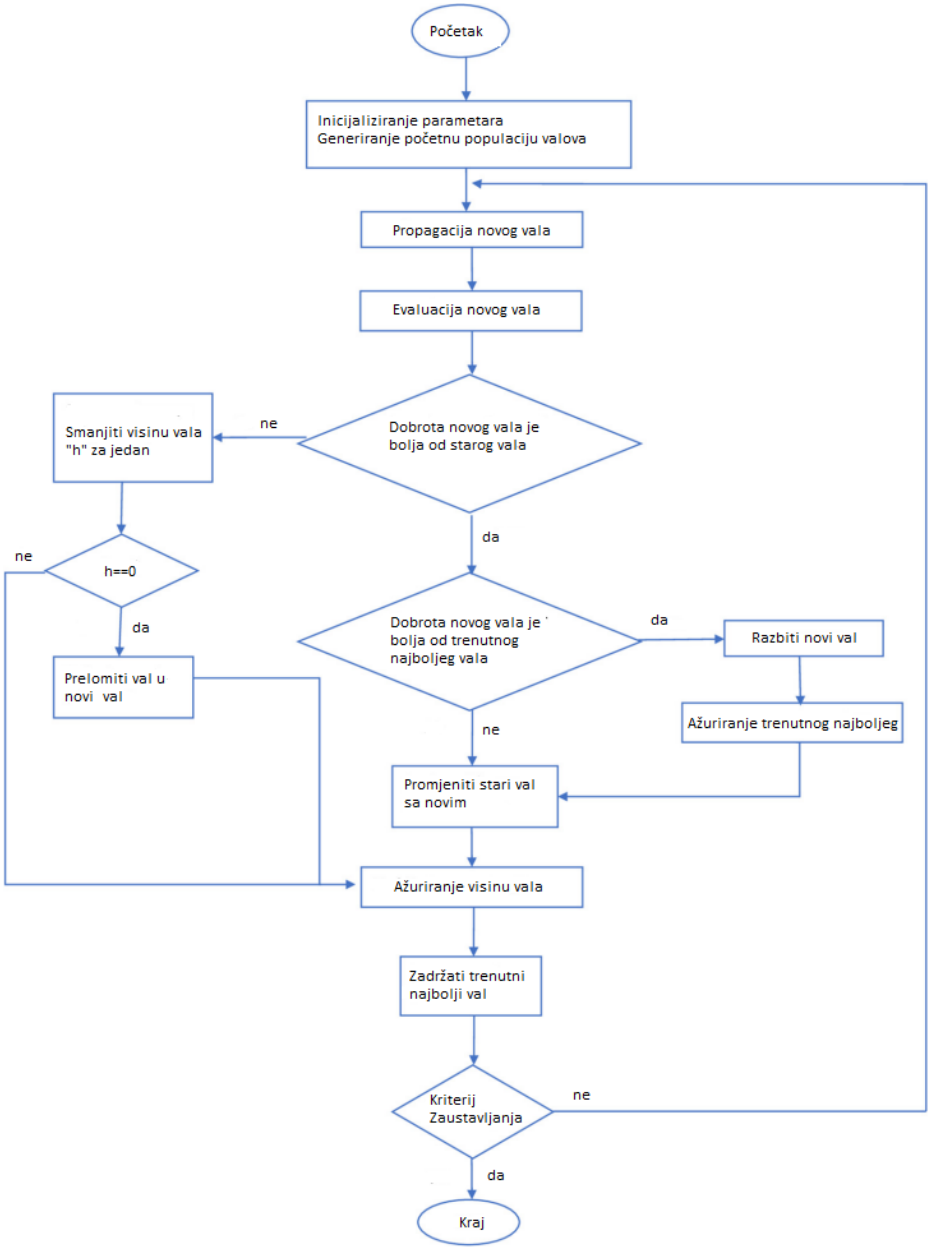
Umjetni imunološki sustavi (eng. Artificial Immune Systems, skraćeno AIS) uzimaju svoju inspiraciju iz teorijske imunologije i promatranih imunoloških funkcija, principa i modela [21]. Oni se mogu smatrati još jednom algoritamskom varijacijom evolucijskih algoritama [22]. Kada se primjenjuju u optimizaciji, antitijela su primarna rješenja koja su se iterativno razvila ponavljanjem operatora kloniranja, mutacije i selekcije. Antigen predstavlja ciljnu funkciju, a dobra rješenja pohranjuju se u memorijsku ćeliju. Gotovo sve metaheuristike temeljene na AIS-u ovise o principima klonske selekcije, kao što su algoritmi negativne selekcije (eng. Negative Selection Algorithms) [23], algoritam klonske selekcije (eng. Clonal Selection Algorithm) [24], optimizacijska verzija umjetne imunološke mreže (eng. optimization version of Artificial Immune Network) [25].

Prikaz Metaheuristike utemeljene na biologiji kroz primjere:

- **Genetski algoritam:** Princip "opstanak najjačih" polazna je točka u predstavljanju mehanizma biološke evolucije. Genetski algoritam oponaša proces biološke evolucije kromosoma definirajući sljedeće operatore: selekciju, križanje i mutaciju. Kromosomi se smatraju mogućim rješenjima za određeni problem i ocjenjuju se prema njihovoj prikladnosti (dobroti). Odabir roditelja važan je proces za stvaranje novih rješenja. Postoje različite metode odabira, kao što su odabir kotača ruleta (eng. roulette wheel selection, skraćeno RWS), odabir turnira (eng. the tournament selection, skraćeno TO), odabir linearnog ranga (eng. the linear rank selection, skrećeno LRS), odabir temeljen na fenotipu (mogućnosti) (eng. the truncation selection, skrećeno TRS) itd.

- **Optimizacija roja čestica** [26]**:** PSO je bio nadahnut ponašanjem životinja u društvu, poput jata riba, roja pčela ili jata ptica. Može se smatrati poluevolucijskim algoritmom inteligencije roja, tj., kao i kod evolucijskog algoritma, skup rješenja odabire se nasumično i procjenjuje kako bi se utvrdilo najbolje rješenje i ponovilo iste procese za određeni broj puta. Za razliku od =procesom pretraživanja, koji također ima brzinu i sjećanje na svoje najbolje pozicije koje je do tad imao.

- **Optimizacija vodenih valova (eng. Water Waves Optimization, WWO)** [27]**:** Optimizacija vodenih valova je nova metaheuristika koju je predložio Zheng [27]. Osnovna ideja WWO-a je simulirati teoriju valova u plitkoj vodi. U WWO, svako moguće rješenje predstavljeno je kao val, a proces pretraživanja percipira se kao gibanje valova, uključujući širenje, lom i refrakcija. Na slici 3. može se vidjeti dijagram toka optimizaciju valova.



Slika 3. Dijagram toka optimizaciju vodenih voda. [12]

- **Algoritam klonske selekcije:**

  Burnet prvi je put predstavio teoriju klonske selekcije 1959. godine kako bi govorio o osnovnom odgovoru adaptivnog imunološkog sustava (limfocita) za antigene [28]. Potvrđuje da će se razmnožavati samo one stanice koje su sposobne identificirati antigen, dok one koje ne identificiraju antigen „odumiru“. CLONALG je optimizacijska verzija klonske selekcije budući da prirodni imunološki sustav može imati više raznih ili kontradikcijskih ciljeva, i nema razloga za razvoj optimalnog odgovora.

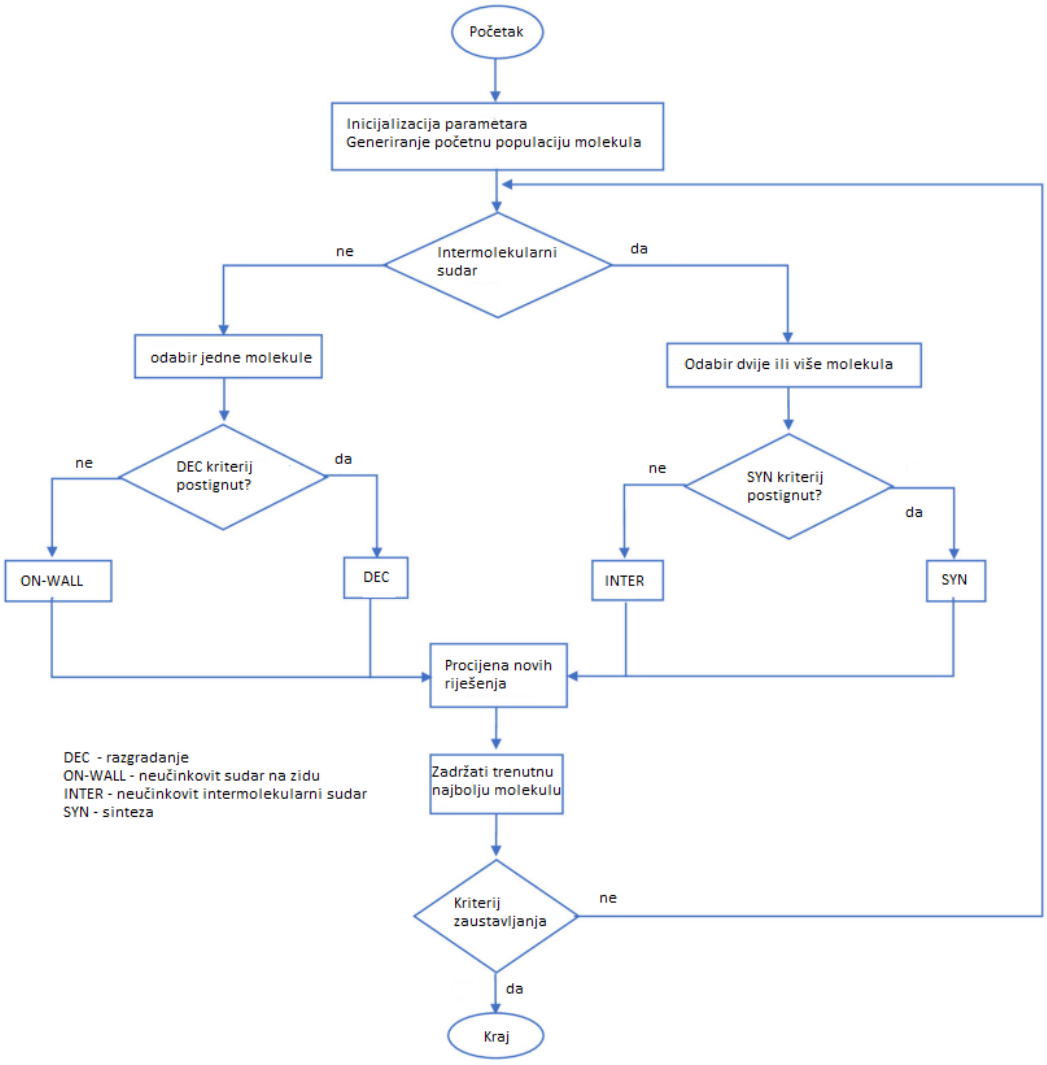
3.1.1.2 Metaheuristika utemeljena na kemiji

Prikaz Metaheuristike utemeljene na kemiji predstavljen je kroz sljedeće primjere:

Optimizacija kemijskih reakcija : Lam i Li predložili su metodu koja oponaša ponašanje interakcije molekula u kemijskoj reakciji kako bi se postiglo stabilno stanje niske energije [29]. U CRO-u, molekula je primarno rješenje, a svaka molekula ima dvije vrste energije: potencijalnu energiju (PE) i kinetičku energiju (KE). Na slici XX može se vidjeti dijagram toka za optimizacijski algoritam kemijskih reakcija. Na slici 4. može se vidjeti dijagram toka za optimizaciju kemijskih reakcija.

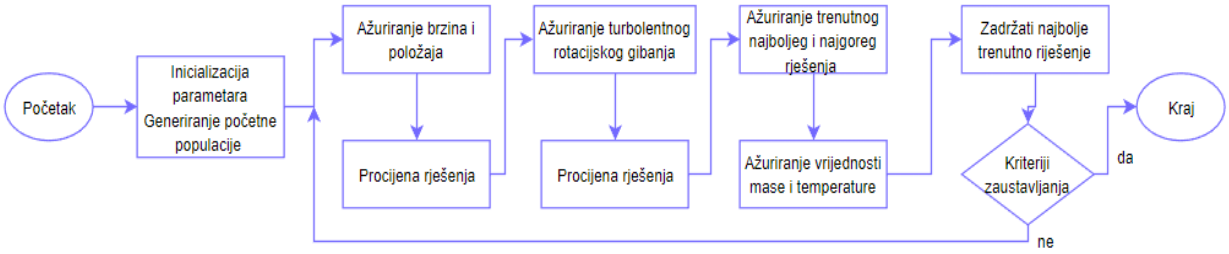
Potencijalna energija odgovara ciljnoj funkciji molekule, a kinetička označava toleranciju za promjenu u lošiju strukturu. Dakle, da bi se molekula Θ zamijenila novom molekulom $\acute{\Theta}$, mora biti ispinjen slijedeći uvjet:

$$PE_{\Theta} + KE_{\Theta} \geq PE_{\acute{\Theta}}$$



Slika 4. *Dijagram toka za optimizacije kemijskih reakcija* [12]

Optimizacija Brownovog kretanja plinova: Inspiracija za optimizaciju Brownovog kretanja plina [30] preuzeta je iz gibanja suspendiranih čestica tekućine (turbulentno rotacijsko i Brownovo gibanje). U GBMO-u svako primarno rješenje predstavljeno je molekulom koja ima četiri karakteritike: položaj, masu, brzinu i radijus turbulencije. Molekule se kreću prema cilju u odnosu na Brownovo kretanje i procjenjuju se prema njegovom položaju. Osim toga, temperatura okoline je dominantan parametar za kontrolu kinetičke energije i brzine molekula i kompromis između istraživanja i eksploatacije, tj. visoka temperatura povećava kinetičku energiju i brzinu molekula i povećava istraživanje, dok niska temperatura smanjuje slučajno Brownovo kretanje i povećava eksploataciju. Suprotno tome, turbulentno rotacijsko gibanje molekula izvodi eksploataciju na visokoj temperaturi i istraživanje na niskoj temperaturi. Na slici 5. može se vidjeti dijagram toka za Brownovo kretanje plinova.



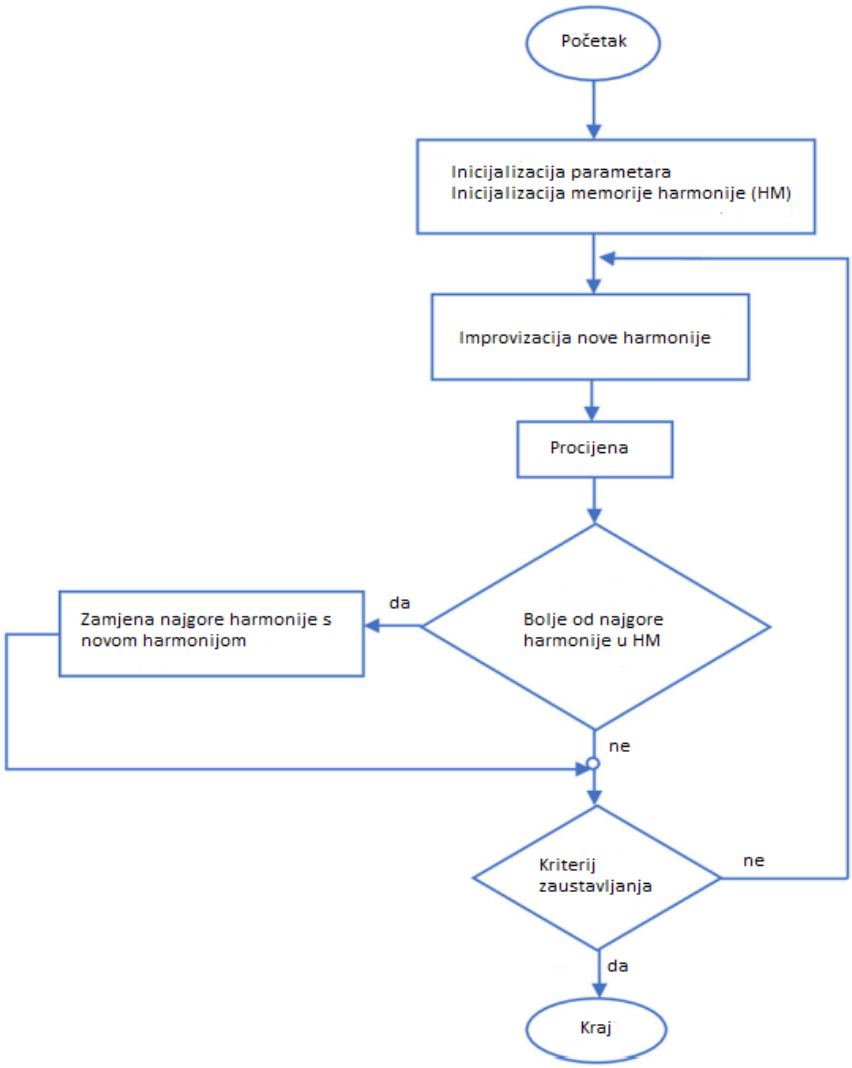
Slika 5. Dijagram toka za optimizaciju Brownovog kretanja plinova [12]

3.1.1.3 Metaheuristika temeljena na glazbi

Prikaz Metaheuristike temeljene na glazbi predstavljen je kroz sljedeće primjere :

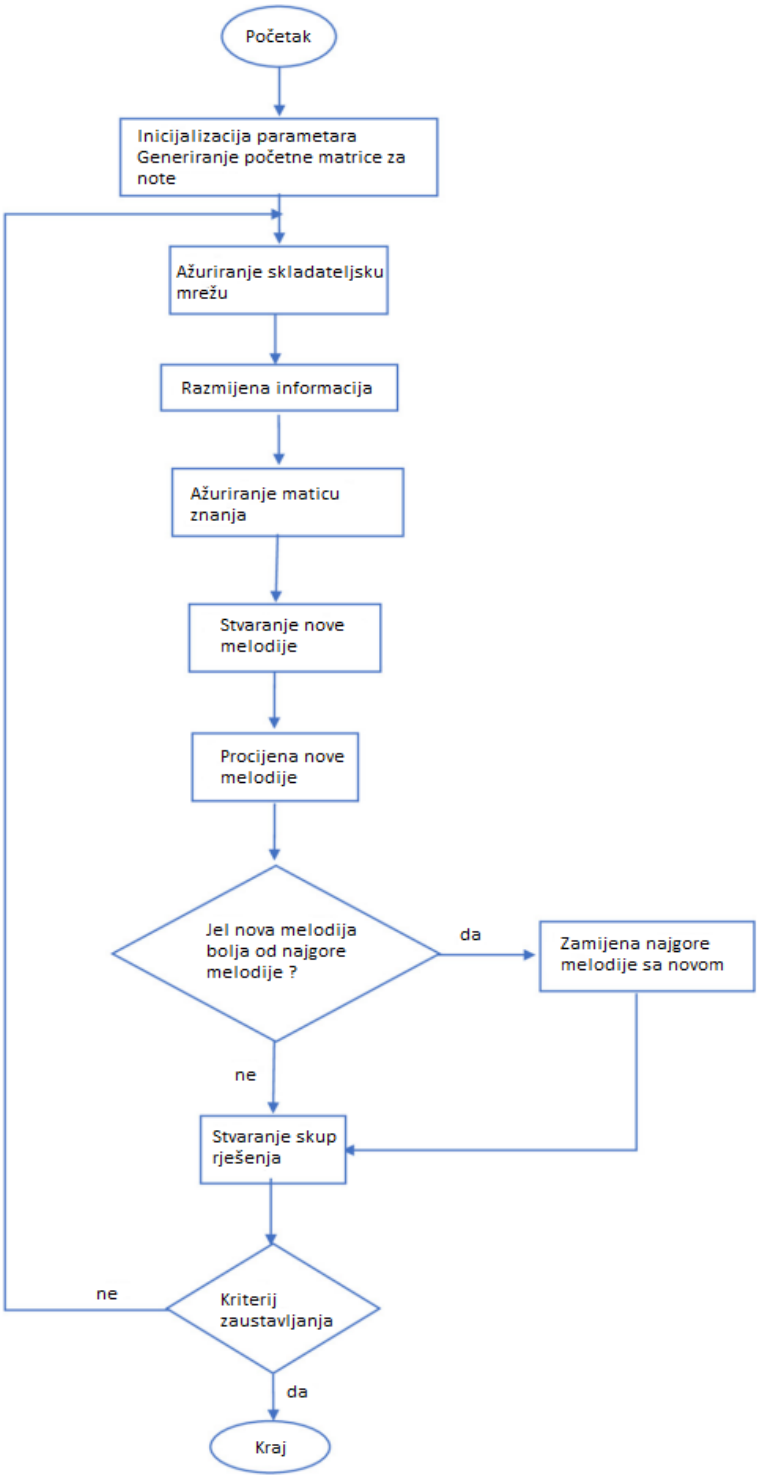
Optimizaciski algotiram za pretraživanje harmonije: Geem [31] je predložio algoritam za pretraživanje harmonije, koji oponaša proces glazbene improvizacije. U glazbenoj improvizaciji svaki instrumentalist svira bilo koji ton unutar mogućeg raspona, što zajedno čini jedan vektor harmonije**.** Ako svi tonovi tvore divnu harmoniju, to će se iskustvo zadržati u sjećanju svakog svirača, a prilika za obradu dobre harmonije povećat će se sljedeći put. U algoritmu HS svaka primarna odluka je vektor. Ako sva rješenja vektora imaju

dobru harmoniju (visoku kondiciju), to se iskustvo pohranjuje u memoriju svake varijable, a mogućnost donošenja dobre odluke također se povećava sljedeći put. Na slici 6. može se vidjeti dijagram toka za algoritam pretraživanje harmonije.



Slika 6. Dijagram toka za pretraživanje harmonije [12]

Metoda glazbene kompozicije: Metoda glazbene kompozicije [32] je simulacija glazbeničkog društva u kojem skladatelji razmjenjuju iskustva međusobno kako bi stvorili glazbenu kompoziciju. Kao i kod pretraživanja harmonije, glavna ideja je tvoriti dobro glazbeno djelo koje uključuje neke već ostvarene skladateljeve kompozicije. Razlika je da metoda glazbene kompozicije slijedi društveni sustav koji uključuje razmjenu informacija i metodologiju učenja. Na slici 7. može se vidjeti dijagram toka metode glazbene kompozicije.



Slika 7. Dijagram toka Metode glazbene kompozicije [12]

3.1.1.4 Matematička metaheuristika

Prikaz Matematičke metaheuristike predstavljen je kroz sljedeće primjere:

- **Osnovni algoritam optimizacije :**

Salem [33] je predložio osnovni algoritam optimizacije, koji ovisi o kombinaciji osnovnih aritmetičkih operacija (+, −, ×, ÷). U prvoj fazi optimizacije nasumično se generira početni skup rješenja i određuje se parametar pomaka. Zatim se procjenjuje svako rješenje i izračunavaju se četiri moguća rješenja pomoću sljedećih formula:

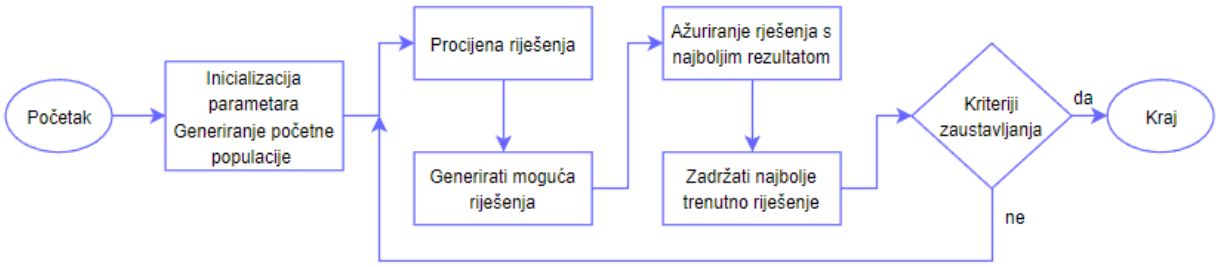
$$x_i^{+}(j) = x_i(j) + \Delta \quad (1)$$

$$x_i^{-}(j) = x_i(j) - \Delta \quad (2)$$

$$x_i^{\times}(j) = x_i(j) \times \Delta \quad (3)$$

$$x_i^{\div}(j) = x_i(j) \div \Delta \quad (4)$$

Gdje je $x_i(j)$ predhodno rijesenje, Δ je parametar pomaka i sve izračunate vrijednosti rješenja unutar određenog raspona. Zatim se procjenjuju četiri moguća rješenja i odabire se najbolje kao novo primarno rješenje. Na slici 8. može se vidjeti dijagram toka za osnovnog optimizacijskog algoritma.



Slika 8. Dijagram toka Osnovnog optimizacijskog algoritma [12]

- **Sinusno-Kosinusni algoritam :** Mirjalili [34] je uveo algoritam, koji je svoje ime dobio po sinus i kosinus jednadžbama. Poput metaheuristike temeljene na populaciji, algoritam započinje slučajnim generiranjem početne populacije i zadržavanjem trenutnog najboljeg rješenja. U procesu pretraživanja koristi se sljedeća jednadžba za ažuriranje lokacija mogućih rješenja u odnosu na trenutno najbolje rješenje:

$$x_i^{(t+1)} = \begin{cases} x_i^t + \left(\delta_1 \times \sin(\delta_2) \times |\delta_3 b_i^t - x_i^t|\right), & if\ \delta_4 < 0.5 \\ x_i^t + \left(\delta_1 \times cos(\delta_2) \times |\delta_3 b_i^t - x_i^t|\right), & if\ \delta_4 \geq 0.5 \end{cases} \quad (5)$$

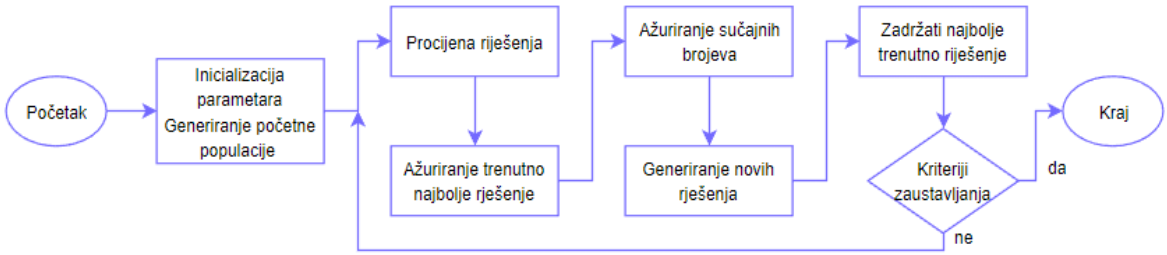
gdje su x_i^t i b_i^t prethodna pozicija trenutnog rješenja i pozicija najboljeg rješenja u i-toj dimenziji na t-toj iteraciji, $\delta_{(1\ldots4)}$ su različiti nasumični brojevi u rasponu

od [0, 1]. Uz to, na svakoj iteraciji, rasponi sinus i kosinus funkcije ažuriraju se radi veće upotrebe prostora za pretraživanje kako slijedi:

$$\delta_1 = c + t\frac{c}{T}$$

gdje je c konstanta, a T je maksimalni broj iteracija.

Važno je napomenuti da su nasumični brojevi $\delta_{(1\ldots4)}$ dominantni u Sinusnom-Kosinusnom algoritmu, tj. $\delta_1$određuje smjer kretanja, δ_2 određuje dužinu pokreta, δ_3 predstavlja nasumičnu važnost za nasumično određivanje odredišta, a δ_4 prebacuje između sinus i kosinus funkcija. Osim toga, promjena u rasponu sinus i kosinus funkcije ogleda se na ažuriranje položaja rješenja. Drugim riječima, ravnoteža između istraživanja i eksploatacije može se postići promjenom raspona sinus i kosinus funkcije. Na slici 9. može se vidjeti dijagram toka za sinus-kosinus algoritam.



Slika 9. Dijagram toka za Sinus-Kosinus algoritam [12]

3.1.1.5 Metaheuristika utemeljena na fizici

Prikaz Metaheuristike utemeljene na fizici predstavljen je kroz sljedeće primjere:

- **Simulirano kaljenje:**

  Metropolis [35] je predložio Simulirano kaljenje za simulaciju hlađenja materijala u toplinskoj kupelji. Nakon toga, Kirkpatrick i suradinici [6] primijenili su ovu ideju na probleme optimizacije. Ime je dobio po procesu fizičkog kaljenja. Dakle, kod početnog stanja kaljenja, algoritam je blag i može prijeći na lošije rješenje. U svakoj iteraciji algoritam postaje stroži za dobivanje boljeg rješenja u svakom koraku.
- **Algoritam gravitacijskog pretraživanja:** Algoritam gravitacijskog pretraživanja je nadahnut Newtonovim zakonima gravitacije i kretanja. U

primarnom rješenju smatra se da objekt ima položaj, inercijalnu masu, aktivnu gravitacijsku masu i pasivnu gravitacijsku masu. Položaji predmeta predstavljaju rješenja, a njihova prikladnost mjeri se njihovim masama. Kao i kod optimizacija roja čestica (PSO) ažuriranje rješenja ovisi o brzini. Objekti s većom produktivnošću i većom gravitacijskom masom imaju veliki radijus učinkovite privlačnosti, a time i veći intenzitet privlačnosti. Dakle, predmeti imaju tendenciju da se kreću prema boljim rješenjima zbog same iste gravitacije.

3.1.1.6 Društvena i sportska metaheuristika

Prikaz Društvene i sportske metaheuristike predstavljen je kroz sljedeće primjere:

- **Optimizacija utemeljena na nastavnom učenju:** Optimizacija utemeljena na nastavnom učenju simulira utjecaj [36] učitelja na učenike u procesu klasičnog učenja u školi, tj. učitelj (primarno rješenje) dijeli svoje znanje s učenicima (skup rješenja), a njegova kvaliteta podučavanja utječe na ocjene učenika (vrijednost dobrote). Proces učenja u TLBO podijeljen je u dvije glavne faze: faza učitelja i faza učenika. U prvom koraku odabire se najbolje rješenje za učitelja i izračunava se prosjek položaja učenika i pomiče prema položaju učitelja. Tada se pozicija novog studenta izračunava kao:

$$x_i^{(t+1)} = x_i^t + r(x^* - TF.\bar{x})$$

gdje je x_i^t prethodna i-ta pozicija studenta, r je nasumičan broj između 0 i 1, x^* je trenutačna pozicija učitelja, $\bar{x}$ srednja vrijednost trenutne populacije, a TF koeficijent učenja koji se izračunava nasumično. U drugoj fazi, učenik i povećava svoje znanje interakcijom s drugim studentom j koji je slučajno odabran. Dva studenta koji se uspoređuju u skladu s tri pravila:

1. Ako su oba rješenja izvediva, tada se odabire rješenje s najboljom vrijednošću dobrote.
2. Ako je jedno rješenje izvedivo, a drugo neizvedivo, tada se odabire izvedivo rješenje.
3. Ako oba rješenja nisu moguća, odabire se rješenje koje ima minimalno kršenje ograničenja izvedivosti.

Znanje učenika i mijenja se ako je znanje odabranog učenika j bilo bolje.

- **Algoritam ligaškog prvenstva (League Championship Algorithm, LCA):** Algoritam ligaškog prvenstva [37] simulira sportsko prvenstvo, gdje nekoliko momčadi igra u ligi nekoliko sezona, tj., kao i u stvarnom sportskom prvenstvu, nekoliko timova (odluka) sudjeluje u ligi (skup odluka) i natječe se u parovima. Utakmica se zatim analizira analizom snage / slabosti / mogućnosti / prijetnje. Pobjednički tim određuje se prema dobroj kondiciji igrača momčadi i snazi igre (vrijednost dobrote). Tada će svaka momčad poboljšati svoje performanse promjenom stila igre i preoblikovanjem svojih igrača. Turniri se nastavljaju nekoliko tjedana prije do kraja sezone. Za analizu utakmice postoji 6 potrebnih pretpostavki:
  1. Ovisno o „snazi igre“ određuje se pobjednički tim.
  2. Jedan rezultat turnira ne daje jednoznačno predviđanje snage igre određene momčadi.
  3. Svaka momčad može se natjecati s bilo kojom drugom momčadi s istom vjerojatnošću.
  4. Izjednačenja se zanemaruju, uzima se u obzir samo rezultat pobjede ili poraza.
  5. Kada se dvije momčadi natječu, svaka snaga koja podržava pobjedu jedne ima odgovarajuću slabost koja je uzrokovala poraz druge.
  6. Momčad se usredotočuje na sljedeću utakmicu, ne obraćajući pažnju na ostale naredne utakmice. Pretvorba se vrši samo na temelju rezultata prethodnog tjedna.

3.1.2 Ne-metaforna metaheuristika

3.1.2.1 Tabu pretraga (Tabu search, TS)

Tabu pretraživanje je uvedeno sa strane Glovera i McMillana [38] koji u prvi put upotrijebili izraz "metaheuristika". Osim toga, tabu pretraga se zbog iteracijskog pretraživanja može svrstati u evolucijske algoritme. Glavna ideja tabu pretrage je zabrana ponovnog posjećivanja već posjećenog područja pretraživanja radi promoviranja istraživanja, tj. kako bi se istražio prostor rješenja i izbjeglo zatvaranje u lokalni optimum (odnosno lokalni minimum i maksimum). Tabu pretraga postavlja bilo koju proceduru za lokalnu pretragu. Dvije glavne značajke uključene u tabu pretraživanje su adaptivna memorija i odzivno istraživanje. Prva (tabu lista) čuva povijest prošlih izvršenih radnji tijekom postupka pretraživanja kako bi se izbjeglo ponavljanje u ciklusima. Drugi koristi prvu proceduru kako bi se proces pretraživanja

usredotočio na dobre regije i najbolje rješenje za veće pojačano istraživanje obećavajućih novih regija za daljnje istraživanje.

3.1.2.2 Pretraga promjenjivih susjedstva (Variable neighborhood search, VNS)

Pretraga promjenjivih susjedstva [39] je stohastička metaheuristika koja koristi jednostavnu lokalnu pretragu i više susjedstva koja se izmjenjuju. Osnove pretrage promjenjivih susjedstva se temelji na sljedeće tri pretpostavke:

1. Lokalni optimum za jedno susjedstvo ne mora biti isti kao i u drugom.
2. Globalni optimum je lokalni optimum unutar svih mogućih susjedstva.
3. Za mnoge probleme lokalni optimumi su blizu jednim drugome, mogu se nalaziti u istome ili različitim susjedstvima.

Posljednja činjenica dobivena je empirijskim promatranjima. Međutim, to podrazumijeva da lokalna optimalna rješenja mogu sadržavati neke korisne informacije o globalnom optimumu.

Osnovna ideja VNS-a je sustavno ili nasumično istražiti više susjedstva tijekom traženja optimalnog (ili blizu optimalnog) rješenja. Pretraga promjenjivih susjedstva se temelji na naizmjeničnoj primjeni dva postupka poboljšanja. Prvo je „protresanje“, koje se radi kako bi se izašlo iz odgovarajućeg lokalnog minimuma. Uz to, primjenjuje se jednostavno lokalno pretraživanje za dobivanje lokalnog optimalnog od ovih susjeda, ili neki napredniji postupci koji istražuju nekoliko susjedskih struktura. Drugi korak je "promjena susjedstva", u kojem će VNS istražiti nasumično izbrisana susjedstva trenutnog rješenja i krenuti od tih susjedstva do novih, ako i samo ako se dolazi do poboljšanja. Na taj se način čuvaju dobre varijable i dobivaju se obećavajući susjedi.

3.1.2.3 Djelomična optimizacija metaheuristike pod posebnim intensifikacijskim uvjetma (Partial optimization metaheuristic under special intensification conditions, POPMUSIC)

Taillard i Voss uveli su POPMUSIC [40] kao metaheuristiku za korištenje kod velikih prostora za pretraživanje. Drugim riječima, POPMUSIC je lokalno pretraživanje koje može optimizirati velike probleme optimizacije (posebno kombinatorne probleme optimizacije) dijeljenjem zadatka na pod-zadatke. Temeljni pristup POPMUSIC-a je dijeljenje problema S u n manjih dijelova $(s_1, s_2, \dots, s_n)$ a zatim formulirati pod-problem R koji se sastoji od odabranog dijela

(glavni dio) i $r < n$ susjednih dijelova prema zadanoj udaljenosti. Nakon formuliranja pod-problema, riješit će se pomoću metaheuristike ili preciznog algoritma. Ti se postupci ponavljaju sve dok se ne riješe pod-zadaci.

3.2 PRIRODOM INSPIRIRANI METAHEURISTIČKI ALGORITMI

Pretraživanje je ključni koncept umjetne inteligencije. Općenito, prostori pretraživanja praktičnih problema obično su toliko veliki da isključuju mogućnost njihovog nabrajanja. To isključuje upotrebu tradicionalnih metoda koje se temelje na računu i nabrajanju. U tu svrhu pokreću se paradigme računalne inteligencije.

Računalna inteligencija područje je umjetne inteligencije. Istražuje adaptivne mehanizme koji promiču inteligentno ponašanje u izazovnim okruženjima. Za razliku od umijetne inteligencije (eng. artificial intelligence, skraćeno AI), koji se oslanja na znanje stečeno ljudskim iskustvom, računalna inteligencija ovisi o prikupljenim numeričkim podacima. Uključuje skup računalnih paradigmi nadahnutih prirodom. Glavne teme u računalnoj inteligenciji uključuju neuronske mreže za prepoznavanje uzoraka, nejasne sustave za rasuđivanje pod neizvjesnošću i evolucijsko računanje za traženje stohastičke optimizacije.

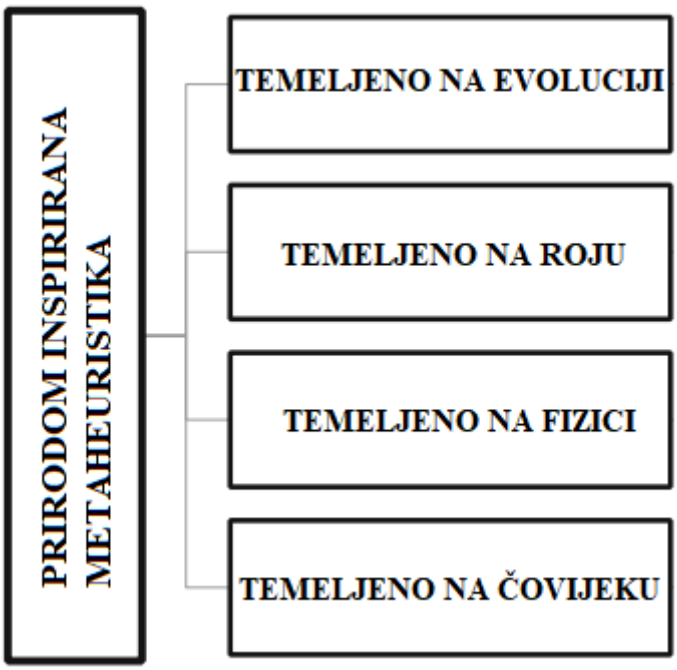
Priroda je glavni izvor inspiracije za nove računalne paradigme. Promjene u prirodi, od mikroskopskih razmjera do ekoloških razmjera, mogu se smatrati proračunima. Prirodni procesi uvijek postižu optimalnu ravnotežu. Takve analogije mogu se koristiti za pronalaženje korisnih rješenja za pretraživanje i optimizaciju.

Od bakterija do ljudi, biološka bića imaju društvenu interakciju u rasponu od altruističke suradnje do sukoba. Inteligencija roja posuđuje ideju kolektivnog ponašanja biološke populacije. Zajedničko rješavanje problema, pristup je u kojem se određeni cilj postiže suradnjom skupine autonomnih učesnika. Mehanizmi suradnje uobičajeni su u paradigmama računanja temeljenih na agentima, bilo biološkim ili ne. Kooperativno ponašanje nadahnulo je istraživače u biologiji, ekonomiji i multiagentnim sustavima. Ovaj se pristup temelji na ideji da se nagrađuje provedba određenih strategija.

3.2.1 Početak razvoja prirodom inspiriranih metaheurističkih algoritama

Riječ "priroda" odnosi se na mnoge pojave uočene u fizičkom svijetu. To uključuje gotovo sve što percipiraju osjetila, pa čak i neke stvari koje se ne percipiraju jednostavno. Priroda je savršen primjer adaptivnog rješavanja problema tj. prilagođava samu sebe da bi se došlo do rješenja; nebrojeno je puta pokazala kako se mnogi različiti problemi mogu riješiti primjenom optimalne strategije koja odgovara svakom pojedinom prirodnom fenomenu. Mnogi istraživači širom svijeta bili su fascinirani načinom na koji se priroda može prilagoditi tako lako u različitim situacijama, a godinama su pokušavali oponašati ove intrigantne sheme rješavanja problema kako bi razvili alate s aplikacijama u stvarnom svijetu.

Zapravo, tijekom posljednja dva desetljeća priroda je služila kao najvažniji izvor inspiracije u razvoju metaheuristike. Kao rezultat toga, pojavila se potpuno nova klasa/tehnika optimizacije u obliku takozvanih algoritama optimizacije ispiriranih prirodom. Ove metode (koje se često nazivaju algoritmima inspiriranih biologijom) posebna su vrsta metaheuristike razvijene s jedinom svrhom: simulirati biološki ili fizički fenomen za rješavanje problema optimizacije. Ovisno o izvoru ispiracije, metaheuristika inspirana prirodom može se podijeliti u četiri glavne kategorije: metode temeljene na evoluciji, temeljene na roju, temeljene na fizici i utemeljene na čovjeku kao što je prikazano na sljedečoj slici. [41]



Slika 10. Klasifikacija prirodom inspiriranih metaheuristickih algoritma [42]

Metode temeljene na evoluciji razvijaju se iz zakona prirodne evolucije. Od ovih metoda najpopularniji je, bez sumnje, pristup genetskim algoritmima koji oponaša evoluciju [43]. Ostale poznate metode grupirane u ovoj kategoriji uključuju evolucijsku strategiju [44], diferencijalnu evoluciju [45] i genetsko programiranje [46]. S druge strane, metode temeljene na rojevima osmišljene su tako da oponašaju socijalno i kolektivno ponašanje koje pokazuju skupine životinja (poput ptica, insekata, riba i drugih), najpopularnijih od njih je optimizacija roja čestica koja je opisana ranije u ovom radu. Osim toga, postoje algoritmi temeljeni na fizici koji su dizajnirani s ciljem oponašanja zakona fizike koji se promatraju u našem svemiru, kao što su simulirano kaljenje i algoritam gravitacijskog pretraživanja. Konačno, postoje i algoritmi koji se temelje na ljudima. Ovakve metode nadahnute prirodom jedinstvene su po tome što inspiraciju crpe iz nekoliko pojava koje su obično povezane s ponašanjem, načinom života ili percepcijom ljudi, a jedna od najpoznatijih metoda objašnjena u literaturi je Optimizacijski algoritam za pretraživanje harmonije [31].

Većina metoda inspirirana prirodom modelirana je kao algoritam temeljen na populaciji, u kojima skupina nasumično generiranih sredstava za pretraživanje (koja se često nazivaju pojedincima), istražuje različita moguća rješenja primjenom određenog skupa pravila izvedenih iz nekog određenog prirodnog fenomena. Ova vrsta razvojnog okvira nudi važne prednosti kako u interakciji između pojedinaca, što pridonosi povećanju znanja o različitim rješenjima, tako i raznolikosti populacije, što je važan aspekt osiguranja da algoritam može učinkovito istražiti prostor dizajna, kao i prevladati lokalne optimume [47]. Zahvaljujući tome i mnogim drugim prepoznatljivim svojstvima, metaheuristike inspirirane prirodom postale su popularan izbor među istraživačima. Kao rezultat toga, literatura koja se odnosi na algoritme optimizacije inspirirane prirodom i njihove primjene za rješavanje složenih problema optimizacije postala je izuzetno velika, a stotine novih radova objavljuju se svake godine.

3.2.2 Generalni razvojni okvir za prirodom inspiriranu metaheuristiku

Uz nekoliko iznimaka, većina prirodom inspirirane metaheuristike, koje su trenutno prikazane u literaturi, modelirane su kao algoritmi temeljeni na populaciji, što implicira da ukupni razvojni okvir koji koristi većina ovih metoda ostaje gotovo identičan, bez obzira na prirodni fenomen na kojem se temelji algoritam [48].

Tipično, prvi korak algoritma inspiriranih prirodom, uključuje definiranje skupa N nasumično inicijaliziranih rješenja $\boldsymbol{X} = \{\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mathrm{N} \,\}$ (obično nazvano populacijom) tako da:

$$\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}] \tag{1}$$

gdje elementi $x_{i,n}$ predstavljaju varijable odluke (parametre) koji se odnose na zadani problem optimizacije, a d označava dimenziju (broj varijabli rješenja) ciljnog prostora rješenja. S pregledom na optimizaciju, svaki skup parametra $\mathbf{x}_i \in \mathbf{X}$ (zvan *pojedinac*) smatra se primarnim rješenjem navedenog problema optimizacije. Svakom od ovih rješenja dodijeljena je i odgovarajuća vrijednost kvalitete (ili dobrote) koja je povezana s ciljnom funkcijom $f(\cdot)$ koja opisuje problem optimizacije, tako da je:

$$f_i = f(\mathbf{x}_i) \tag{2}$$

Metode inspirirane prirodom obično slijede shemu iterativnog pretraživanja u kojoj se nova primarna rješenja generiraju mijenjanjem trenutno dostupnih *pojedinaca*, a to se postiže primjenom nekih prethodno postavljenih kriterija (obično razvijenih na temelju promatranog prirodnog fenomena). U većini slučajeva ovaj se postupak može ilustrirati sljedećom jednadžbom:

$$\mathbf{x}_i' = \mathbf{x}_i + \Delta\mathbf{x}_i \tag{3}$$

gdje $\mathbf{x}_i'$ označava primarno rješenje generirano zbrajanjem navedenih vektora ažuriranja $\Delta\mathbf{x}_i$ do $\mathbf{x}_i$. Vrijedno je napomenuti da vrijednost (i) koju uzima vektor ažuriranja $\Delta\mathbf{x}_i$ ovisi o određenim operaterima koje koristi svaki pojedinačni algoritam.

Konačno, većina algoritama inspirirana prirodom uključuje neku vrstu selekcijskog postupka u kojem se novo generirana rješenja uspoređuju s onima u trenutnoj populaciji $\mathbf{X}^k$ (gdje k označava trenutnu iteraciju) u smislu kvalitete rješenja, obično s ciljem odabira najboljih pojedinaca među njima. Kao rezultat ovog procesa, generira se novi skup rješenja $\mathbf{X}^{k+1} = \{\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \ldots, \mathbf{x}_n^{k+1}\}$ što odgovara slijedećoj iteraciji (tz. generaciji) '$k+1$'.

Cijeli se ovaj postupak iterativno ponavlja dok se ne ispune određeni kriteriji za zaustavljanje (npr. ako se dostigne maksimalni broj iteracija). Jednom kada se to dogodi, najbolje rješenje pronađeno algoritmom postavljeno je kao najbolja aproksimacija globalnog optimuma [48].

3.2.3 Najvažnije referentne funkcije

Referentne (kontrolne) funkcije su funkcije koje se najčešće koriste za uspoređivanje metaheurističkih algoritma. Genetički algoritmi u velikoj se mjeri oslanjaju na generatore nasumičnih brojeva. Uz to, svaki od glavnih genetskih operatora koji se koriste u jednostavnom GA (križanje, mutacija) koristi „nasumični odabir“ u određenoj mjeri. Mnogi istraživači iz ove grane znanosti često su koristili različite funkcionalne skupine za proučavanje karakteristika GA, dok se velik dio istraživanja odnosi na određivanje kontrolnih parametara i njihovo potencijalno prilagođavanje. Tu će se prikazati 14. najvažnijih referentnih funkcija.

Name	*Function*	*Limits*
F1	$f_1 = \sum_{i=1}^{2} x_i^2$	$-5.12 \leq x_i \leq 5.12$
F2	$f_2 = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	$-2{,}048 \leq x_i \leq 2{,}048$
F3	$f_3 = \sum_{i=1}^{5} \text{int.}(x_i)$	$-5{,}12 \leq x_i \leq 5{,}12$
F4	$f_4 = \sum_{i=1}^{30} (ix_i^4 + \text{Gauss}(0, 1))$	$-1{,}28 \leq x_i \leq 1{,}28$
F5	$f_5(x_i) = 0.002 + \sum_{j=1}^{25} \left(\frac{1}{j} + \sum_{i=1}^{2} (x_i - a_{ij})^6 \right)$	$-65536 \leq x_i \leq 65536$
F6	$f_6 = 10\,V + \sum_{i=1}^{10} (-x_i \sin(\sqrt{\lvert x_i \rvert})),$ $V = 4189.829101$	$-500 \leq x_i \leq 500$
F7	$f_7 = 20\,A + \sum_{i=1}^{20} (x_i^2 - 10\cos(2\pi x_i)),$ $A = 10$	$-5{,}12 \leq x_i \leq 5{,}12$
F8	$f_8 = 1 + \sum_{i=1}^{10} \left(\frac{x_i^2}{4000} \right) - \prod_{i=1}^{10} \left(\cos\left(\frac{x_i}{\sqrt{i}} \right) \right)$	$-600 \leq x_i \leq 600$

Slika 11. Prvih osam referentnih funkcija za uspoređivanje metaheurističkih algoritama [49]

Prvih pet referentnih funkcija predložio je DeJong. Sve testne značajke odražavaju različite stupnjeve težine [50]. Testne funkcije F1 do F4 su unimodalne (tj. sadrže samo jedan optimum), dok su ostale testne funkcije multimodalne (tj. sadrže mnogo lokalnih optimuma, ali samo jedan globalni optimum).

Sfera (Sphere) [**F1**] je glatka, unimodalna, visoko konveksna, simetrična funkcija.

Rosenbrock [**F2**] funkcija smatra se izazovnim jer ima vrlo uski val. Vrh vala je vrlo oštar i prolazi preko parabole. Algoritmi koji nisu u stanju otkriti dobre puteve nisu dobri u ovom zadatku.

Step [**F3**] je glavna funkcija za probleme koje sadrže ravne površine. Ravne površine su prepreke algoritmima optimizacije jer ne daju nikakve informacije o tome koji je smjer povoljan.

Quartic [**F4**] je jednostavna unimodalna funkcija koja ima šum. Gaussov šum osigurava da algoritam nikada ne dobije istu vrijednost u istoj točki. Algoritmi koji ne rade dobro s ovom testnom značajkom neće dobro raditi s podacima koji imaju šum.

Foxholes [**F5**] je primjer funkcije s mnogim lokalnim optimumima. Mnogi standardni algoritmi za optimizaciju zapnu na prvom vrhu ili nizini na koju naiđu.

Schwefel, Rastrigin, Griewangk [**F6, F7, F8**] su tipični primjeri nelinearnih multimodalnih funkcija. Rastriginova funkcija je prilično veliki izazov za genetske algoritme zbog velikog prostora pretraživanja i velikog broja lokalnih minimuma. Schwefelova funkcija nešto je jednostavnija od Rastriginske funkcije, a karakteriziraju je najbolji minimumi koji su postavljeni daleko od globalnog optimuma.

Većina algoritama teško se približava minimumu tih funkcija, jer se vjerojatnost napretka brzo smanjuje kako se približava minimumu. Preostale funkcije (prikazane na slici ispod) koje su odabrane kao referentne funkcije za genetičke algoritme preuzete su iz područja matematičkog programiranja i uglavnom pokrivaju područje nelinearnih programskih problema [51].

Name	*Function definition*	*Dimensions*
F9	$f_i = \sum_{j=2}^{n}(j-1)x_j t_i^{(j-2)} - \sum_{j=1}^{n}(x_j t_i^{(j-1)})^2 - 1$ where $t_i = i/29$, $1 \le i \le 29$ $f_{30}(x) = x_1, f_{31}(x) = x_2 - x_1^2 - 1$ Standard starting point: $x_0 = (0, \ldots, 0)$	$2 \le n \le 31$, $m = 31$
F10	$f_{2i-1}(x) = 10(x_{2i} - x_{(i-1)}^2)$ $f_{2i}(x) = 1 - x_{(2i-1)}$ Standard starting point: $x_0 = (\xi_j)$ where $\xi_{2j-1} = -1.2, \quad \xi_{2j} = 1$	n variable but even $m = n$
F11	$f_1(x) = x_1 - 0.2$ $f_i(x) = \alpha^{(1/2)}\left(\exp\left[\frac{x_i}{10}\right] + \exp\left[\frac{x_{(i-1)}}{10}\right] - y_i\right), \quad 2 \le i \le n$ $f_i(x) = \alpha^{(1/2)}\left(\exp\left[\frac{x_{(i-n+1)}}{10}\right] - \exp\left[\frac{-1}{10}\right]\right), \quad n \le i \le 2n$ $f_{2n}(x) = \left(\sum_{j=1}^{n}(n-j+1)x_j^2\right) - 1$ where $\alpha = 10^5, y_i = \exp\left[\frac{i}{10}\right] + \exp\left[\frac{(i-1)}{10}\right]$ Standard starting point: $x_0 = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$	n variable $m = 2n$
F12	$f_1(x) = 10^4 x_1 x_2 - 1$ $f_2(x) = \exp[-x_1] + \exp[-x_2] - 1.0001$ Standard starting point: $x_0 = (0, 1)$	$n = 2$, $m = 2$
F13	$f_1(x) = \exp\left[\frac{-\lvert y_i\ mix_2\rvert^{(x_1)}}{x_1}\right] - t_i$ $t_i = i/100$ $y_i = 25 + (-501\mathrm{n}(t_i))^{(2/3)}$ Standard starting point: $x_0 = (5, 2.5, 0.15)$	$n = 3$, $n \le m \le 100$
F14	$f_{(4i-3)}(x) = x_{(4i-3)} + 10x_{(4i-2)}$ $f_{(4i-2)}(x) = 5^{(1/2)}(x_{(4i-1)} - x_{4i})$ $f_{(4i-1)}(x) = (x_{(4i-2)} - 2x_{(4i-1)})^2$ $f_{(4i)}(x) = 10^{(1/2)}(x_{(4i-3)} - x_{(4i)})^2$ Standard starting point: $x_0 = (\xi_j)$ where $\xi_{4j-3} = 3, \quad \xi_{4j-2} = -1, \quad \xi_{4j-1} = 0, \quad \xi_{4j} = 1$	n variable but a multiple of 4 $m = n$

Slika 12. F9-F14 referentne funkcije za uspoređivanje metaheurističkih algoritama [49]

4. ANALIZA PRIRODOM INSPIRIRANIH METAHEURISTIČKIH ALGORITAMA

U ovom poglavlju su prikazani neki od najnovijih prirodom inspiriranih metaheurističkih algoritama koji su proizašli iz ove grane znanosti u zadnjih nekoliko godina.

4.1 ALGORITAM ZA OPTIMIZACIJU LOVA NA JELENE

Rad [52] koji je napisao G. Brayama, predlaže novi metaheuristički algoritam utemeljen na optimizaciji lova na jelene (eng. Deer Hunting Optimization Algorithm, skraćeno DHOA). Iako se postupci lovaca razlikuju, način na koji napadaju jelena/srnu temelji se na strategiji lova koju razvijaju.

4.1.1 Generalno

U prirodi opstanak je uvjetovan društvenim ponašanjem koje pomaže u obavljanju različitih zadataka. Pojedinci formiraju skupine, čopore, stada, jata itd., za različite svrhe kao što su lov, obrana itd.

Ovaj rad predlaže novi metaheuristički algoritam koji je inspiriran lovom na jelene od strane skupine lovaca. Da bi lovili jelena, lovci ga okružuju i kreću prema njemu na temelju nekoliko strategija. Strategije uključuju uzimanje u obzir različitih parametara kao što su kut vjetra, položaj jelena i slično. Suradnja između lovaca još je jedan važan kriterij koji lov čini učinkovitim. Napokon, postižu cilj zasnovan na položaju vođe i nasljednika. Strategija lova ovisi o premještanju dva lovca (vođa i nasljednik) na njihove najbolje položaje. U skladu s tim, svaki lovac ažurira svoj položaj dok ne dođe do jelena.

4.1.2 Inspiracija

Iako je glavni cilj predloženog algoritma pronalaženje optimalnog položaja za lov na jelene, potrebno je proučiti ponašanje jelena/srne. Generalno, oni imaju određene karakteristike koje otežavaju lov predatorima. Jedna od tih karakteristika je vid, koji je pet puta bolji od ljudskog. Međutim, imaju problema s percepcijom crvene i zelene boje. Jelen, može primijetiti čak i lagano kretanje, a biolozi kažu da bjelorepi jelen ima periferni vid u rasponu od 250 do 270 stupnjeva. To pomaže jelenu da otkrije kretanje lovca, ali samo ispod horizonta. Budući da je jelenu teško uhvatiti kretanje iznad horizonta, jelen navodno ne može primijetiti lovca koji stoji visoko na stablu. Osjetilo njuha bjelorepog jelena je izvrsno, šezdeset puta je bolje od ljudskog. Može osjetiti najmanju opasnost, zahvaljujući mirisnim senzorima. Osjetivši bilo kakvu

opasnost, upozorava ostale jelene sa teškim topotom i glasnim rikanjem. Jedna od značajnih vještina koje sluh jelena posjeduje je njegova sposobnost da pokupi zvukove ultra visoke frekvencije koji nisu dostupne ljudima. Uši jelena su poput satelitskih antena koji se okreću kako bi registrirali različite zvukove.

4.1.3 Algoritam

Pomoću idućih koraka matematički se opisuje optimizacijski algoritam za lov na jelene.

4.1.3.1 Inicijalizacija populacije

Glavni korak algoritma je inicijalizacija populacije lovaca, kao što je prikazano u nastavku:

$$Y = \{Y_1, Y_2, \ldots, Y_n\}; \qquad 1 < j \leq n \tag{1}$$

gdje je n ukupan broj lovaca, a predstavljaju rješenja u populaciji Y.

4.1.3.2 Inicijalizacija parametara

Nakon inicijalizacije populacije, inicijalizira se kut vjetra i kut položaja jelena, koji su važni parametri u određivanju najboljih položaja lovaca. Budući da se prostor za pretraživanje smatra krugom, kut vjetra odgovara opsegu kruga:

$$\theta_i = 2\pi r \tag{2}$$

gdje je r nasumični broj između [0, 1], a i predstavlja trenutnu iteraciju. Dok se kut položaja jelena postavlja pomoću:

$$\phi_i = \theta + \pi \tag{3}$$

gdje je θ kut vjetra.

4.1.3.3 Propagacija položaja

Budući da je položaj optimalnog prostora u početku nepoznat, algoritam uzima primarno rješenje blizu optimuma, koji se određuje na temelju funkcije dobrote (eng. fitness function), kao najbolje rješenje. Ovdje se razmatraju dva rješenja, a to su položaj vođe (leader) koji je predstavljen kao Y^{lead}, što je prva najbolja pozicija lovca, i položaj nasljednika (eng. successor), Y^{succ}, koji je položaj sljedećeg lovca.

1. **Propagacija putem pozicije vođe:** Jednom kada se utvrde najbolji položaji, svaki pojedinac u populaciji pokušava postići najbolji položaj i tako započinje postupak

ažuriranja položaja. U skladu s tim, ponašanje okoline modelira se kako je navedeno u sljedećoj jednadžbi,

$$Y_{i+1} = Y^{\text{lead}} - X \cdot p \cdot \left| L \times Y^{\text{lead}} - Y_i \right| \quad (4)$$

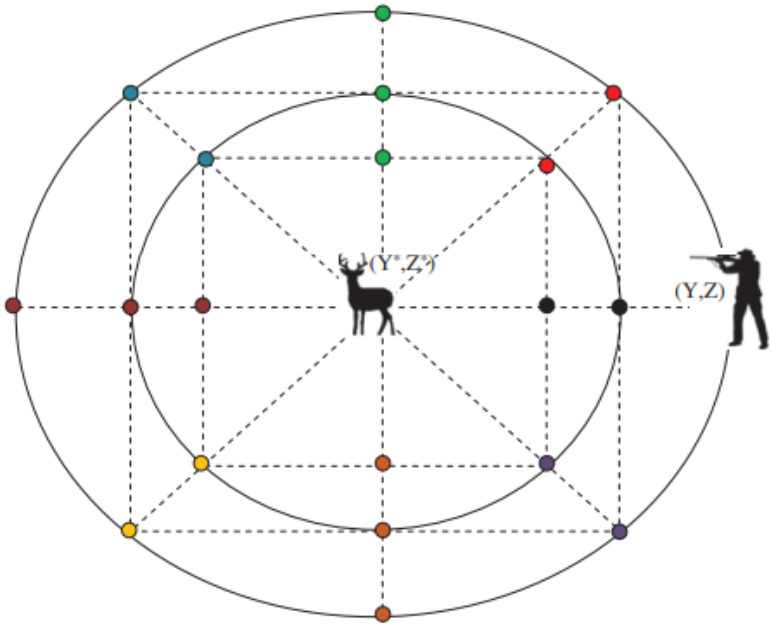
gdje je Y_i položaj na trenutnoj iteraciji, Y_{i+1} je položaj na sljedećoj iteraciji, X i L su koeficijenti vektora, a p je nasumični broj dobiven uzimajući u obzir brzinu vjetra čija je vrijednost između 0 i 2. Koeficijenti vektora mjere se kao,

$$X = \frac{1}{4}\log(i + \frac{1}{i_{max}})b \quad (5)$$

$$L = 2 \cdot c \quad (6)$$

gdje je i_{max} maksimalna iteracija, b je parametar koji ima vrijednost između -1 i 1, a c je nasumični broj u intervalu [0, 1].

Ovo je ponašanje prikazano na Slici 13. gdje (Y, Z) predstavlja početni položaj lovca koji se može ažurirati prema položaju jelena. S promjenom vrijednosti X i L postigne se najbolji položaj agenta pretraživanja (Y^*, Z^*). Svaki lovac kreće se prema položaju vođe ako je moguće. Međutim, lovac ostaje u trenutnom položaju, ako se vođa ne kreće ili se krivo kreće. Ažuriranja položaja prati jednadžbu koeficijenta vektora X samo ako $p < 1$, što znači da se pojedinac može proizvoljno kretati u bilo kojem smjeru, bez obzira na kut položaja. Tako da koristeći prijašnje jednadžbe lovac može ažurirati svoje mjesto na bilo kojem nasumičnom mjestu u prostoru.



Slika 13. Prikaz ažuriranja položaja lovca [52]

2. **Propagacija kroz kut položaja:** Da bi se proširio prostor za pretraživanje, koncept se proširuje uzimajući u obzir kut položaja. Izračun kuta potreban je za određivanje položaja lovca na takvo mjesto da ga jelen ne primijeti i stoga je postupak lova učinkovit. Kut vida jelena ili srne izračunava se kao,

$$a_i = \frac{\pi}{8} r \tag{8}$$

Na temelju razlike između kuta vjetra i kuta vida jelena izračunava se parametar koji pomaže u ažuriranju kuta pozicioniranja.

$$d_i = \theta_i - a_i \tag{9}$$

gdje je θ kut vjetra. Zatim se kut položaja ažurira za sljedeću iteraciju.

$$\phi_{i+1} = \phi_i + d_i \tag{10}$$

Uzimajući u obzir kut položaja, ažuriranje položaja vrši se kako je navedeno u jednadžbi ispod,

$$Y_{i+1} = Y^{\text{lead}} - p \cdot \left|\cos(v) \times Y^{\text{lead}} - Y_i\right| \tag{11}$$

gdje $A = \phi_{i+1}$, Y_i^* predstavlja najbolji položaj, a p nasumični broj. Uglavnom, položaj pojedinca gotovo je suprotan kutu položaja jelena, tako da lovac nije u vidnom polju jelena.

3. **Propagacija putem pozicije nasljednika :** U fazi istraživanja, ista ideja u ponašanju okoline može se usvojiti prilagođavanjem vektora L. Budući da u početku pretpostavljamo nasumično pretraživanje, vrijednost vektora L je manja od 1. Dakle, ažuriranje položaja temelji se na položaju nasljednika, a ne na prvom dobivenom najboljem rješenju. To omogućuje globalno pretraživanje predstavljeno sljedećom jednadžbom,

$$Y_{i+1} = Y^{\text{succ}} - X \cdot p \cdot |L \times Y^{\text{succ}} - Y_i| \tag{12}$$

gdje je Y^{succ} pozicija nasljednika od agenta pretraživanja iz trenutne populacije.

Počevši od nasumične inicijalizacije različitih rješenja, algoritam ažurira položaj agenata u svakoj iteraciji na temelju dobivenog najboljeg rješenja. Kada $|L| < 1$, agent za pretraživanje odabire se nasumično, dok je najbolje rješenje odabrano je kad $|L| \geq 1$ za ažuriranje položaja agenata. Stoga, adaptivnom varijacijom vektora L, predloženi se algoritam prebacuje između faza istraživanja i eksploatacije. Štoviše, postoje samo dva parametra koja je potrebno prilagoditi, X i L, što algoritmu dodaje prednosti.

4.1.4 Rezultat algoritma s obzirom na kontrolne funkcije

Predloženi algoritam za optimizaciju lova na jelene uspoređuje se s 39 kontrolnih funkcija, a njegova numerička kompetencija analizira se usporedbom nekoliko metaheurističkih algoritama koji su prije navedeni. Prva 33 mjerila temelje se na tradicionalnim referentnim funkcijama. S druge strane, zabilježene su rezidualne kontrolne funkcije, koje su klasične funkcije koje koriste mnogi istraživači. Ove referentne funkcije imaju veliku sposobnost uspoređivanja ostvarenih rezultata s rezultatima trenutne metaheuristike, unatoč jednostavnosti. Budući da su svi metaheuristički algoritmi stohastičke prirode, postizanje optimalnog rezultata nije lak zadatak. Stoga se predloženi algoritam DHOA i konvencionalni algoritmi izvršavaju 30 puta za svaku kontrolnu funkciju i dobivaju statističko izvješće. Funkcije cilja, izvedene iz svih kontrolnih funkcija DHOA, jasno ukazuju na to da su operativne sposobnosti DHOA-a konkurentne u usporedbi s tradicionalnim metaheurističkim algoritmima. Uz to, broj lokalnih minimuma eksponencijalno se povećava s populacijom. Stoga se istraživačka sposobnost algoritama može pravilno procijeniti pomoću ove vrste testnih problema. Ovaj algoritam može lako postići globalni optimum zahvaljujući kombiniranom istraživačkom mehanizmu. Očito je da algoritam izvrsno radi na rješavanju svih testnih problema i vrlo je konkurentan i razuman u usporedbi s drugim algoritmima. To određuje da DHOA može u velikoj mjeri uravnotežiti faze istraživanja i eksploatacije. Na slici 14. nalazi se pseudokod za optimizacijski algoritam za lov na jelene.

Input: Inicijalizacija populacije Y
Output: Prvo najbolje rješenje Y^{lead} i drugo najbolje rješenje Y^{succ}
Početak
  dok $(i < i_{max})$
    za svako riješenje u populaciji
      Izračunati dobrotu za svako rješenje
      Ažuriraj a, d, A, p, X, L i b
      ako je $(p < 1)$
        ako je $(|L| \geq 1)$
          Ažuriraj poziciju pojedinca putem jednadže (4)
        Inače
          Ažuriraj poziciju pojedinca putem jednadže (12)
      Inače
          Ažuriraj poziciju pojedinca putem jednadže (11)
    Izračunati dobrotu za svako rješenje
    Ažurirat Y^{lead} i Y^{succ}
    $i = i + 1$
  vratiti Y^{lead}
Kraj

Slika 14. Pseudokod za optimizacijski algoritam za lov na jelene [52]

4.2 METAHEURISTICKI ALGORITAM TEMELJEN NA PLAVOM MAJMUNU

Ovaj rad [53] je napisali su Maha Mahmood i Belal Al-Khateeb, predstavlja studiju algoritma Plavog majmuna (eng. Blue monkey, skraćeno BM), koji je metaheuristički algoritam optimizacije koji se temelji na ponašanje plavih majmuna u prirodi. BM algoritam određuje koliko ima muškaraca u jednoj skupini majmuna. Izvan sezone razmnožavanja, u skupinama plavih majmuna postoji samo jedan odrasli dominantni mužjak, kao i kod ostalih šumskih vrsta majmuna.

4.2.1 Inspiracija

Cercopithecus mitis (tj. plavi majmun) se udružuje s Cercopithecus ascanius (tj. crveno repati majmun) za dodatnu zaštitu. Socijalni sustav plavih majmuna u osnovni je matrijarhalni, jer mužjaci odlaze čim postanu spolno zreli [54]. Plavi majmuni nisu poput ostalih vrsta majmuna. Obično žive u društvenim sustavima u kojima dominiraju žene, što znači da žene ostaju u svojim rodnim skupinama. S druge strane, mužjaci napuštaju svoje skupine čim dosegnu fazu zrelosti. Većina skupina plavih majmuna ima mnogo ženki i mladunaca, ali samo jednog mužjaka. Zbog toga se otežava miješanje različitih gena unutar jedne zajednice. Mužjaci napuštaju skupinu u drugu skupinu kad dostignu zrelost, ali pronalazak druge skupine može

potrajati, pa se mogu činiti kao samostalni mužjaci. U grani društvenih znanosti, plavi majmuni nemaju vrlo snažne instinkte [55]. Vrijeme društvenih interakcija je kratko, obično se događalo tijekom zajedničke igre i njege. Osim toga, postoji interakcija između mladunčad, njihovih majki i drugih odraslih ženka u grupi. To dovodi do toga da se ta djeca obično ne približavaju muškim majmunima u istoj grupi. Mlade ženke brinu se o mladuncima, nose ih i štite. Ova navika uči mladunčad interakciji sa svim majmunima kasnije u životu.

4.2.2 Algoritam

Algoritam je podijeljen u dva dijela: podjela skupina i ažuriranje položaja.

4.2.2.1 Podjela skupina

BM algoritamski program oponaša ponašanje plavog majmuna. Da bi se simulirale takve interakcije, svaku skupinu jedinice majmuna trebalo bi je postaviti iznad područja pretraživanja. Kao što je ranije spomenuto, plavi majmuni, kada se podijele u timove, počinju tražiti mjesta za hranu na velikim udaljenostima i počinju pretragu za jakim majmunom nad kojim mogu dominirati. Mužjaci plavog majmuna praktički ne komuniciraju s mladuncima. Zbog teritorijalne prirode, mladi mužjaci moraju izaći što je brže moguće kako bi postali uspješniji. Sukobit će se s dominantnim mužjakom iz druge obitelji. U slučaju da uspiju pobijediti ovog mužjaka, oni mogu biti vođe te obitelji, tako da mogu ponuditi zalihe hrane, mjesto za boravak i socijalizaciju za mlade mužjake. Zbog toga skupine plavih majmuna imaju jednog mužjaka i veliki broj ženki i mladunaca.

4.2.2.2 Ažuriranje položaja

Ažuriranje položaja za svakog plavog majmuna u grupi ovisi o najboljem položaju plavog majmuna u toj skupini, ovo se ponašanje određuje sljedećim jednadžbama:

$$R_{i+1} = (0.7 \cdot R_i) + p\,(W_{lead} - W_i)\,(X_{lead} - X_i) \quad (1)$$

$$X_{i+1} = X_i + R_{i+1} \cdot p \quad (2)$$

gdje je R faktor snage majmuna (i za različite iteracije), W_{lead} je težina vođe, W_i je težina majmuna pri čemu su sve težine nasumični brojevi između [4, 6], X je položaj majmuna (i za različite iteracije), X_{lead} je položaj vođe, a p je nasumični broj između [0, 1]. Položaj se mora ažurirati na svakoj iteraciji.

Pored toga, kako bi se ažurirala mladunčad plavih majmuna, koriste se sljedeće jednadžbe:

$$R_{i+1}^{ch} = \left(0.7 \cdot R_i^{ch}\right) + p\left(W_{lead}^{ch} - W_i^{ch}\right)\left(X_{lead}^{ch} - X_i^{ch}\right) \quad (3)$$

$$X_{i+1}^{ch} = X_i^{ch} + R_{i+1}^{ch} \cdot p \quad (4)$$

gdje R^{ch} fakotor snage majmuna (i za različite iteracije), W_{lead}^{ch} je težina vođe, W_i^{ch} je težina majmuna pri čemu su sve težine nasumični brojevi između [4, 6], X^{ch} je položaj majmuna (i za različite iteracije), X_{lead}^{ch} je položaj vođe, a p je nasumični broj između [0, 1]. Položaj se mora ažurirati na svakoj iteraciji.

4.2.3 Rezultat algoritma s obzirom na kontrolne funkcije

BM algoritam testira se i ocjenjuje pomoću 43 kontrolne funkcije. Prvih 18 funkcija su jednostavne i tradicionalne značajke koje koriste mnogi istraživači. Te su funkcije odabrane kako bi mogle usporediti rezultate s onima nekih dobro poznatih metaheurističkih algoritama. Općenito, korištene testne funkcije su funkcije minimiziranja koje su ili unimodalne ili multimodalne kontrolne funkcije. Predloženi BM algoritam izveden je 30 puta za svaku od kontrolnih funkcija.

Rezultati BM algoritma su nadmašili druge odabrane algoritme u većini odabranih 18 kontrolnih funkcija. Zbog toga su unimodalne funkcije prikladne za analizu eksploatacije prema određenim spomenutim operaterima.

Mnogi lokalni optimumi mogu se naći u multimodalnim funkcijama, čiji se broj eksponencijalno povećava s povećanjem dimenzionalnosti. Stoga su multimodalne funkcije prikladne za provjeru pouzdanosti algoritma istraživanja. Rezultati prikazuju da u 25 kontrolnih funkcija BM algoritam nadmašuje određene prirodno inspirirane metaheurističke algoritme. Na slici 15. nalazi se pseudokod za algoritam temeljen na plavom majmunu.

Input: Inicijalizacija populacije plavog majmuna i mladunaca ($i = 1 \ldots n$)
Inicijalizacija faktora snage R i težina W ($R \in [0, 1]$), ($W \in [4, 6]$)
Output: najbolje riješenje tj. plavi majmun
Početak
Podijela majmune u nasumične timove (T), i mladunce u druge
Izračunati dobrotu za sve mladunce i plave majmune za svaku grupu
Za svaku grupu, izabrere se najbolje i najgore vrijednost dobrote i postavlja se kao trenutno riješenje. Dok se kod mladunaca uzima najbolja vrijednost dobrote
$t = 1$
Dok ($t \leq$ maksimum broj iteracija)
Zamjeniti lošiju vrijednost dobrote u svakoj grupi sa najboljom.
Ažurirati R i X pozicije plavih majmuna u svaku grupu putem jednadžbe (1)
Ažurirati R i X pozicije mladunaca putem jednadžbe (2)
Ažurirati dobortu za sve plave majmune i mladunce
Ažurirati trenutnu primarnu poziciju:
ako je novo riješenje bolje od trenutnog onda Trenutno riješenje =Novo riješenje
$t = t + 1$
završi dok
vratiti optimalnog plavog majmuna
Kraj

Slika 15. Pseudokod za algoritam temeljen na plavom majmunu [53]

4.3 METAHEURISTIČKI ALGORITAM PRETRAGE TEMELJEN NA MEDVJEĐEM NJUHU [56]

U ovom radu (autor Ali Ghasemi-Marzbali) predstavljen je novi metaheuristički algoritam optimizacije nadahnut prirodom; algoritam pretrage temeljen na medvjeđem njuhu (eng. Bear smell search algorithm, skraćeno BSSA), koji uzima u obzir moćne globalne i lokalne operatore pretraživanja.

4.3.1 Generalno

Klasične metode koristile su linearnu strukturu kako bi pronašle najbolje rješenje [57]. Budući da problemi optimizacije imaju mnogo lokalnih optimalnih rješenja i nelinearnu strukturu, oni ne mogu uspješno pronaći najbolje ili globalno rješenje te su zbog toga zarobljeni u lokalnom rješenju.

Priroda pruža istraživačima velike mogućnosti za stvaranje novih umjetnih računalnih algoritama za rješavanje složenih problema koji oponašaju ponašanje životinja, biljaka i ostalog. Prema sveobuhvatnoj studiji ponašanja životinja i njihovih osjetila u prirodi, može se zaključiti da je medvjeđi njuh je jedan najjačih osjetila u prirodi.

Predloženi algoritam simulira dinamičko ponašanje medvjeda, temeljeno na mehanizmu mirisa, i migracijama medvjeda. Medvjed prelazi na tisuće kilometara u potrazi za hranom. U usporedbi s većinom ostalih životinja, medvjedi imaju jači osjet njuha zahvaljujući svojim velikim mirisnim receptorima koje kontroliraju percepciju različitih mirisa. S obzirom da su olfaktivni receptori neuronski model, mogu se pažljivo ispitati i koristiti za optimizaciju. Ovisno o raznim kvalitetama mirisa, medvjed se premješta na sljedeći teritorij.

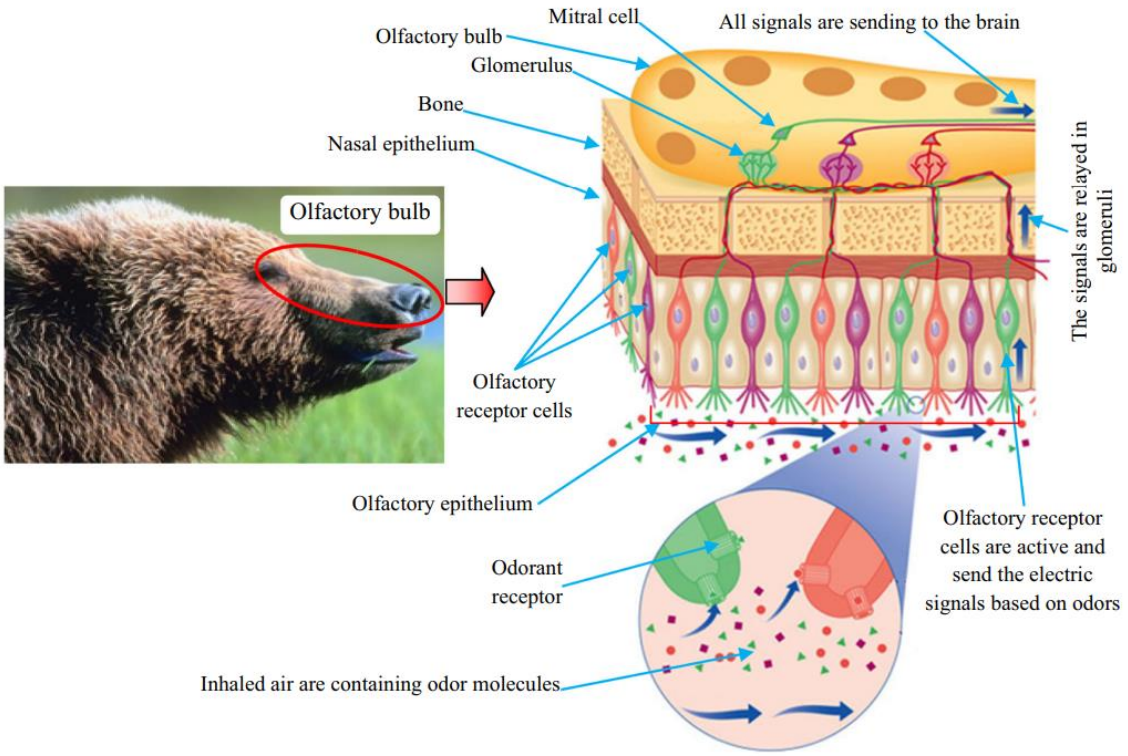
4.3.2 Inspiracija

Predviđanje kvalitete mirisa iz skupa komponenata mirisa, uzimajući u obzir njihovu interakciju, težak je zadatak, ali je relativno jednostavno odrediti kvalitetu mirisa u odnosu na medvjeđe preferencije. Nos (eng. olfactory bulb) sa olfaktivnim receptorima (eng. olfactory receptors) prima mirise i prenosi te informacije duž olfaktivnog živca (eng. olfactory tract) u mozak. Jednostavna struktura osjetila, uvjetuje jednostavnost kortikalnog modela i njegovu ulogu u obradi signala. Kao što je prikazano na donjoj slici, ovaj model treba izračunati omjer glomerularne aktivnosti između komponenata mirisa kao ulaz (input). Zapravo, ulaz predstavlja indeks sličnosti između dobivenih mirisa. Medvjedi se mogu prebaciti na sljedeću poziciju ovisno o njihovim sličnostima. Zbog lakšeg razumijevanja, rad medvjeđeg njuha dijeli se na:

Glomerularni dio (eng. Glomerular part): ovaj se sloj sastoji od nekoliko glomerularnih jedinica koje mjere uzorak glomerularne aktivnosti temeljen na komponenata mirisa (eng. odor components). Izlaz (output) ovog sloja je matrica komponenata mirisa.

Mitralni i granularni dio (eng. Mitral and Granular part): mjesto mitralnih stanica (eng. mitral cell) nalazi se ispod glomerularnog sloja. Svaka od mitralnih i granularnih stanica sadrži nerazgranati dendrit u jednom glomerulu koji ispunjava veći dio glomerula unutar kojeg se nalazi. Nadalje, stanice granula nalaze se ispod sloja mitralnih stanica u debelom sloju. Tijela mitralnih stanica leže u sloju iznad tijela zrnastih stanica. Svaka stanica granule ima gornje dendritičko stablo koje se grana i završava u vanjskom pleksiformnom sloju.

Procijena različitih mirisa (eng. Dissimilarity assessment part): funkcije mozga procjenjuju kvalitetu mirisa čija se živčana aktivnost prenosi mitralnim kanalom u njušni korteks mozga. Stoga se nejednake procijenjene vrijednosti mirisa mogu smatrati metrikom za medvjede, omogućujući im da među dobivenim mirisima odaberu navodni miris ili miris svoje omiljene hrane.



Slika 16. Pregled olfaktornog sustava medvjeda [56]

4.3.3 3.3.3. Matematičko formuliranje

Prvo pretpostavimo da je nos medvjeda upio različite mirise, tako da svaki od njih pokazuje smjer kretanja, jer sve u okolini ima svojstven miris. Miris željene hrane je konačno rješenje i smatra se globalnim rješenjem.

$$O_i = [oc_i^1 \;\; oc_i^2 \ldots \;\; oc_i^j \ldots \; oc_i^k] \tag{1}$$

gdje je i dobiveni miris sa strane medvjeda sa k komponetama ili molekula. Budući da medvjed tijekom disanja prima n mirisa, znači da je početno rješenje matrica.

$$\mathrm{OM} = [O_i]_{n\times k} = \; [oc_i^j]_{n\times k} \tag{2}$$

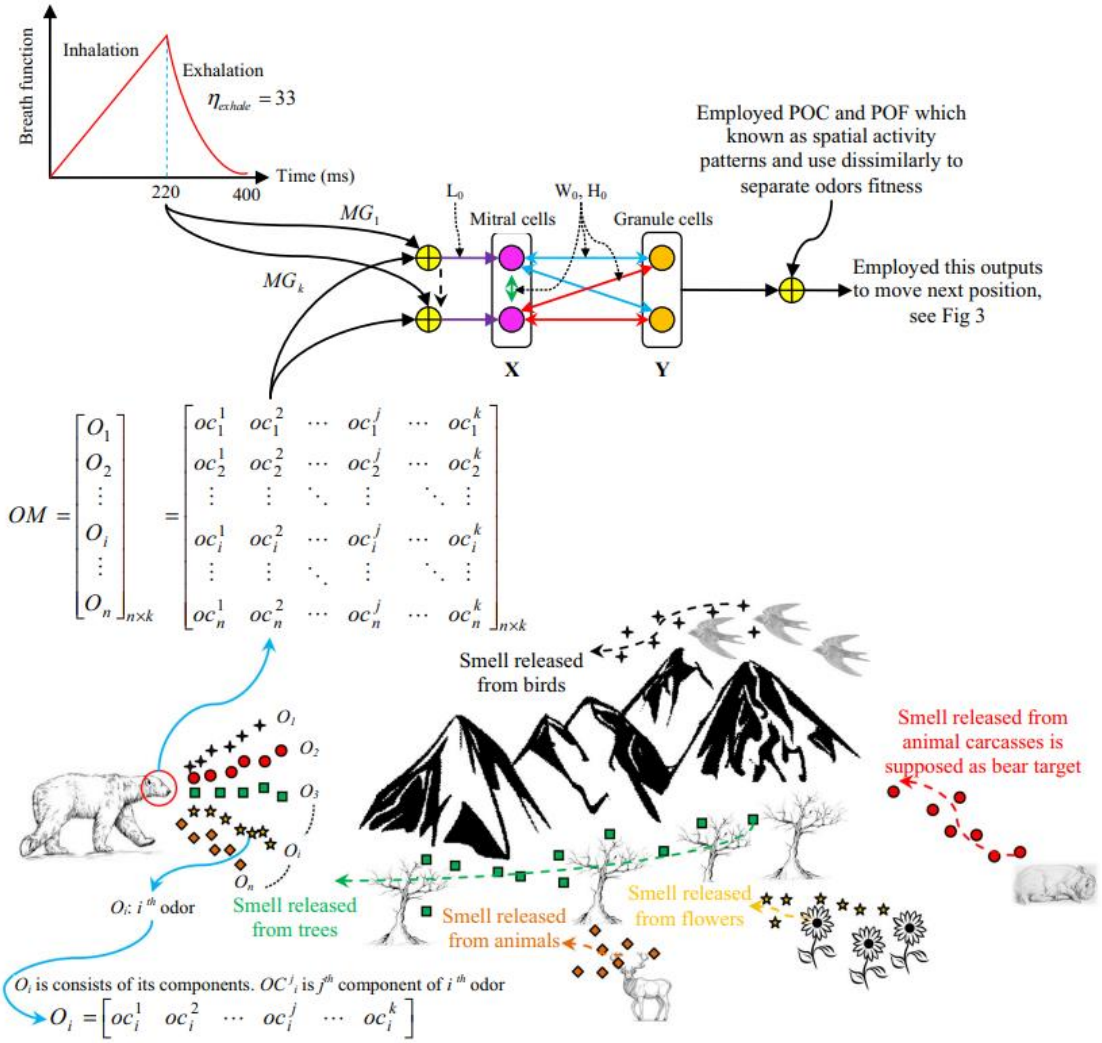
Prema procesu glomerularnog sloja i funkciji disanja u ciklusu, ako DS_i^j označuje j-ti komponenta mirisa u i-tom mirisu. Koristeći matematičko formuliranje [58] dobije se:

$$\mathrm{DS}_i^j = \begin{cases} \mathrm{MG}_i(t - t_{udah}) + DS_i^{t_{udah}}, & t_{udah} \le t \le t_{izdah} \\ \mathrm{DS}_i^{t_{izdah}} \exp\left(\frac{t_{izdah} - t}{\eta_{izdah}}\right), & t_{idah} \le t \end{cases} \tag{3}$$

gdje su t_{izdah}, t_{udah} i η_{izdah} vrijeme izdaha, vrijeme udaha i konstantna vrijednost za vrijeme izdaha. U procesu optimizacije, ukupno vrijeme ciklusa disanja podudara se s duljinom mirisa k i odgovara vremenu izdisaja i udisanja. Komponente mirisa podijeljene su u dvije skupine.

$\mathrm{MG} = \{\mathrm{MG}_1,\ \mathrm{MG}_2, \ldots\ \mathrm{MG}_i, \ldots\ \mathrm{MG}_n\}$ sadrži osjetljivost receptora, kao i apsorpciju i identifikaciju mirisa, a oni imaju ulazni signal za svaki i-ti mitral. Ako je $\mathrm{DS}_i^j = 0$ pokazuje da olfaktorni epitel nema mirisa prije sljedećeg udisanja. Nenegativni MG može se izračunati pomoću [59]:

$$\mathrm{MG}_i(O_i) = \frac{1}{k}\sum_{j=1}^{k} f\left(oc_i^j\right), \qquad f\left(oc_i^j\right) = \begin{cases} 1, & T_1 \le oc_i^j \\ 0, & T_1 > oc_i^j \end{cases} \tag{4}$$



Slika 17. Grafički prikaz rada olfaktornog sustava [56]

gdje k označava dužinu percepcije mirisa za i-ti miris, T_1 je varijabla praga i postavlja se na temelju srednje vrijednosti informacija o mirisu. Sada se ti podaci šalju u mitralni i granularni sloj. Da bi se postigao taj cilj korištene su formule od Li–Hopfield (3) and the Erdi [58] koje imitiraju neuronsku dinamiku koja se javlja u granularnom i mitralnom sloju :

$$f_x(x) = \begin{cases} 0.14 + 0.14\tanh\left(\frac{x-\varphi}{0.14}\right), & x < \varphi \\ 0.14 + 1.4\tanh\left(\frac{x-\varphi}{1.4}\right), & x \geq \varphi \end{cases} \tag{5}$$

$$f_y(y) = \begin{cases} 0.29 + 0.29\tanh\left(\frac{x-\varphi}{0.29}\right), & x < \varphi \\ 0.29 + 2.9\tanh\left(\frac{x-\varphi}{2.9}\right), & x \geq \varphi \end{cases} \tag{6}$$

gdje je φ vrijednost praga. H_0, W_0 i L_0su matrice za sinapričke veze koje označavaju vezu između granularnih i mitralnih stanica, a izračunava se sljedećim formulama [60]:

$$H_0{}_i^j = \begin{cases} \frac{\text{rand()}}{T_h}, & 0 < d_i^j < T_h \\ 0, & T_h < d_i^j \end{cases} \tag{7}$$

$$W_0{}_i^j = \begin{cases} \frac{\text{rand()}}{T_w}, & 0 < d_i^j < T_w \\ 0, & T_w < d_i^j \end{cases} \tag{8}$$

$$L_0{}_i^j = \begin{cases} \frac{\text{rand()}}{T_l}, & 0 < d_i^j < T_l \\ 0, & T_l < d_i^j \end{cases} \tag{9}$$

T_h, T_w i T_l su konstante sinaptičkih veza, rand() je nasumična vrijednost. d_i^j predstavlja razliku između i-tog i g-tog mirisa uzrokovan njihovim kvalitetom, gdje je g miris koji medvjed želi. Drugim riječima, ta se udaljenost određuje između svakog mirisa (lokalno rješenje) i željenog mirisa (globalno rješenje). To pokazuje da se kontrolirani mehanizam zasnovan na globalnom rješenju koristi u procesu optimizacije za poboljšanje rada eksploatacije. Prema gornjim opisima, kada mozak primi sve informacije iz neuronske aktivnosti, započinje proces razdvajanja na temelju procjene različitih mirisa. Ovaj se postupak modelira na temelju Pearsonove korelacije($r = \frac{\sum(x_i-\bar{x})(y_i-\bar{y}))}{\sqrt{\sum(x_i-\bar{x})^2\ \sum(y_i-\bar{y})^2}}$). Dakle, ova točka pomaže medvjedu da odabere najbolji način kako doći do željenog mirisa gdje su vjerojatnost komponente mirisa (eng. Probability odor component, skraćeno POC), vjerojatnost mirisne

dobrote (probability odor fitness, POF) i mirisna dobrota (Odor fitness, OF) određene sljedećim jednadžbama:

$$\mathrm{POC}_i = \frac{O_i}{\max(O_i)} \tag{10}$$

$$\mathrm{POF}_i = \frac{OF_i}{\max(OF_i)} \tag{11}$$

Razlika između dva mirisa može se izračunati pomoću očekivane dobrote mirisa (eng. Expected odor fitness, skraćeno EOF) i udaljenosti komponenta mirisa (eng. Distance odor components, skraćeno DOC) tako da slijedi:

$$\mathrm{DOC}_i = 1 - \frac{\sum_{i=1}^{k}(\mathrm{POC}_j^1 - \mathrm{POC}_j^2)}{\sqrt{\sum_{i=1}^{k}(\mathrm{POC}_j^1 - \mathrm{POC}_j^2)}} \tag{12}$$

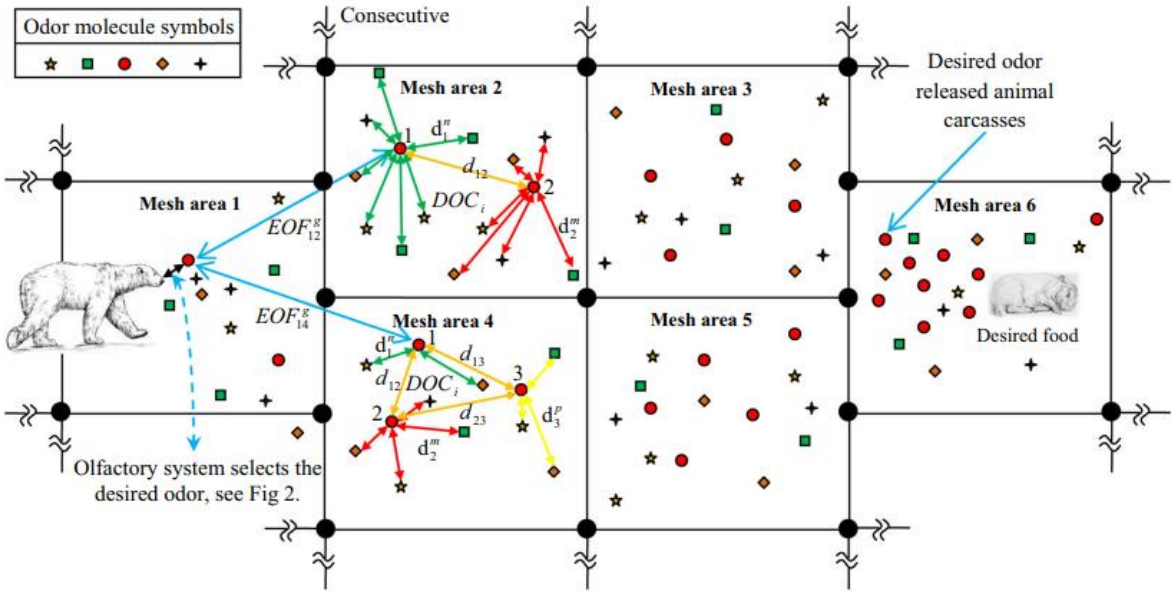
$$\mathrm{EOF}_i = |\mathrm{POF}_i - \mathrm{POF}^g| \tag{13}$$

gdje se g odnosi na globalno riješenje. Gornje jednadžbe diktiraju mogući način kretanja. Zapravo, ovi pokazatelji objašnjavaju vezu između mirisa i željenog položaja (prikazano na donjoj slici). Ilustrira da izlazi (output) iz mozga određuju sljedeći potez, bolje rečeno sljedeći pomak za medvjeda. U svakom području mreže, udaljenost između svih mirisa izračunava se na temelju dva praga:ξ_1, ξ_2 tako se sljedeći mirisi mogu izračunati pomoću:

$$O_{k+1} = \begin{cases} C_{1,i} \times O_k - \mathrm{rand} \times C_{2,i} \times \left(O_k - O_{bolji}\right), & \mathrm{DOC}_i \le \xi_1 \text{ i } \mathrm{EOF}_i \le \xi_2 \\ C_{3,i} \times O_k - \mathrm{rand} \times C_{4,i} \times \left(O_k - O_{bolji}\right), & \text{inače} \end{cases} \tag{14}$$

$$C_{1,i} = -\mathrm{EOF}_i \frac{2-\mathrm{DOC}_i}{\xi_1}, \qquad C_{2,i} = -\mathrm{EOF}_i \frac{2-\mathrm{DOC}_i}{\xi_2}$$

$$C_{3,i} = \mathrm{EOF}_i \frac{2-\mathrm{DOC}_i}{\xi_1}, \qquad C_{4,i} = -\mathrm{EOF}_i \frac{2-\mathrm{DOC}_i}{\xi_2}$$



Slika 18. Objašnjenje općeg koncepta mrežnog mehanizma za pomicanje medvjeda na sljedeći položaj [56]

4.3.4 Rezultat algoritma s obzirom na kontrolne funkcije

Za procijenu algoritama pretrage medvjeđeg njuha koriste se različiti kriteriji koji se temelje na sličnosti njihovih bitnih fizičkih svojstava i oblika (npr. puno lokalnih rješenja, ravnina itd.). Uz to, koristi se nekoliko inženjerskih rješenja kako bi se pokazala izvrsnost u odnosu na ostala rješenja prikazana kroz literaturu. Da bi se imala normalna klasifikacija na temelju relativne sličnosti, korištene su 53 kontrolne funkcije.

Za procjenu i usporedbu performansi po gore spomenutim kontrolnim funkcijama korišteni su popularni optimizacijski algoritmi (npr. genetički algoritam, Optimizacija roja čestica PSO, Algoritam gravitacijskog pretraživanja) i neki koji su potencijalno slični ovom algoritmu (npr. optimizacija sivog vuka, algoritam pretrage za pticu kukavicu, algoritam pretrage za vjevericu). Da bi se napravila poštena usporedba između ovih algoritama, uzimaju se u obzir iste početne uvjete, poput broja stanovništva i broja iteracija. Uspoređujući algoritme za optimizaciju nakon 30 neovisnih pokretanja, dobiju se rezultati o valjanosti BSSA.

Prema statičkim indeksima, može se vidjeti da BSSA može pronaći najbolja ili globalna rješenja za sve kontrolne funkcije. Važno je napomenuti da noviji algoritmi imaju najjaču konkurenciju s BSSA metodom u rješavanju kontrolnih funkcija.

Metaheurističke tehnike optimizacije imaju za cilj uravnotežiti uvjete eksploatacije i istraživanja kako bi se iskoristile sve mogućnosti u procesu pretraživanja. Eksploatacija je koristila neku vrstu nadograđenog operatera koji ima bolje rješenje za poboljšanje konvergencije, dok istraživanje stvara nova rješenja kroz prostor za pretraživanje kako bi se držali podalje od lokalnih rješenja. Nakon računanja i prikaza, BSSA učinkovito održava ravnotežu između mjernih podataka istraživanja i eksploatacije tijekom većeg dijela postupka pretraživanja. Na slici 19. nalazi se pseudokod za algoritam pretage temeljen na medvjeđeg njuhu.

Početak
Input: Broj populacija N, Maximum broj iteracija $Iter_{max}$, gornja i donja granica LB i UB, broj varijabla D.
Output: Nacrt i ispis zeljenih izlaza
Inicijalizacija pocetne nasumične populacije dimenzije $OM = N \times D$ u prostoru pretrage
Procijeniti dobrotu za svaki redak matrice OM
Spreminit najbolji miris kao globalno riješenje O^g
za $Iter = 1$ do $Iter_{max}$
    Izračunati maksimalni izlaz za glomerularnu aktivnost sa jednadžbom (2); MG
    Dobiti DS s obzirom na funkciju udisaja (1)
    Izračunati stanične izlazne funkcije za mitral i granularne stanice sa (4)-(5); $f_x(x)$ i $f_y(y)$
    Odredit H_0, W_0 i L_0 sa jednadžbama (6)-(8)
    Riješiti jednadžbu (3)
    Odrediti vektore POC, POF i OF sa jednadžbama (9)-(10)
    Iztačunati vektore EOF i DOC sa jednadžbama (11)-(12)
    Postavljat pragove ξ_1 i ξ_2
    Izračunati koeficijent C_1 do C_4
    za $i = 1$ do N
        ako zadovoljava kriterije onda
            Generirati novu populaciju temeljen na prvi dio jednadžbe (13)
        inače
            Generirati novu populaciju temeljen na prvi dio jednadžbe (13)
Kraj

Slika 19. Pseudokod za algoritam pretraživanja za miris medvjeda [56]

4.4 METAHEURISTIČKI ALGORITAM INSPIRIRAN JELENIMA

Ovaj rad [61] napisao je Amir Mohammad. Proučava i oponaša ponašanje škotskog jelena kako bi razvio novi algoritam nadahnut prirodom. Glavna inspiracija za ovaj metaheuristički algoritam je neobično ponašanje parenja škotskog Jelena.

4.4.1 Generalno

Slično drugim metaheuristikama koje se temelje na populaciji, algoritam inspiriran jelenima (eng. Red deer algorithm, skraćeno RDA) započinje s početnom populacijom (Red deers, RDs). Podijeljeni su u dvije vrste, na srne i jelene RDs. Osim toga, Harem je skupina srna (ženskih RDs). Sveukupne faze ovog evolucijskog algoritma razmatraju se konkurencijom jelena kako bi dobili Harem kroz rikanje i borbeno ponašanje.

4.4.2 Inspiracija

Škotski jelen (Cervus Elaphus Scoticus) podvrsta je Jelena porijeklom s Britanskih otoka. Od kraja pleistocenske ere populacija jelena postoje u Britaniji, posebno u Škotskoj. U većini slučajeva nalaze se na vrhovima planinskih šuma i močvarnih područja [62]. Populacija Jelena podijeljena je u dvije vrste: jelen (mužjak) i srna (ženka). Jedno od glavnih obilježja ponašanja ove životinje događa se tijekom sezone tjeranja, kada mužjaci glasno i često ričuu.

Slika 20. Primjer škotskog jelena [63]

Srne preferiraju rikanje koji započinje visokom tonom i spušta se prema nižem, ali ne i obratno. Visoka razina rike pozitivno je povezana s reproduktivnim uspjehom i borbenom sposobnošću. Natjecanje između mužjaka za parenje može rezultirati istim obrascima asocijativnog parenja kao i ženski izbor. Nakon što mužjaci postanu glavni u skupinu ženki, nastaje Harem. Zapovjednici brane srne u svom haremu i na svom teritoriju. Dok tijekom jesenskog parenja jeleni ričuu kada se sakupljaju ili štite hareme [64].

Natjecanje mužjaka za čin zapovjednika smatra se predvidljivom borbom. Što se tiče jelena i zapovjednika, jedan mužjak prilazi drugom i postaje istaknut, ričući u njegovom smjeru. Suparnici koji se približavaju obično razmjenjuju riku u trajanju od nekoliko minuta. Ako

dvoboj pređe u drugu fazu, izazivač napreduje i kreće prema zapovjedniku. U većini slučajeva hodaju jedan pored drugog, obično pod pravim kutom. Tijekom hodanja, bilo koji od jelena može izazvati kontakt okrećući se prema protivniku i spuštajući rogove [65]. U skladu s tim, svaki jelen pokušava okrenuti svog protivnika prema padini tako da je niži od njega, tada se pokušaju srušiti. Na kraju pobjednik postaje zapovjednik harema.

Još jedna glavna karakteristika jelena je njihovo rutinsko ponašanje. Rutina jelena vrlo je predvidljiva iz godine u godinu. Sredinom rujna zreli jeleni (5 god iznad) koje su prethodnih 10 mjeseci proveli u društvu sa ostalim jelenima postaju netolerantni jedni prema drugima, te se pojedinci sele u tradicionalna područja, gdje okupljaju i štite skupine ženki (harem).

4.4.3 Algoritam

Slično kao i drugi metaheuristički algoritmi, RDA započinje početnom nasumičnom populacijom. Odabiru se neki od najboljih RD-ova u populaciji i dijele se u jelene i srne. Jeleni su postavljeni da riču, prema jačini i načinu rike su podijeljeni u dvije grupe (obični jeleni i zapovjednici). Nakon toga, zapovjednici i jeleni svakog harema bore se zajedno kako bi preuzeli svoj harem (hareme formiraju zapovjednici). Broj srna u haremima izravno je povezan sa sposobnostima zapovjednika u rikanju i borbenom procesu. S time da se zapovjednici pare s nekoliko ženki u haremima. Ostali jeleni se pare sa preostalim ženkama, ne uzimajući u obzir ograničenja harema.

4.4.4 Generiranje inicijalnog jelena

Cilj optimizacije je pronaći gotovo optimalno ili globalno rješenje u smislu varijabli problema. Ovdje se oblikuje niz vrijednosti varijabli koje treba optimizirati. *RedDeer* je korišten za niz gdje postoji mogućnost rješenja X u prostornom riješenju. Dimenzija ovog rešenja X je N. Znaci *RedDeer* je niz od 1 × N. Jednadžba ispod pokazuje komponente za svaku dimenziju od X.

$$RedDeer = [X_1, X_2, X_3, \ldots, X_N] \tag{1}$$

Vrijednosti funkcije za svakog jelena može se pokazati ovako:

$$Vrijednost = f(RedDeer) = f(X_1, X_2, X_3, \ldots, X_N) \tag{2}$$

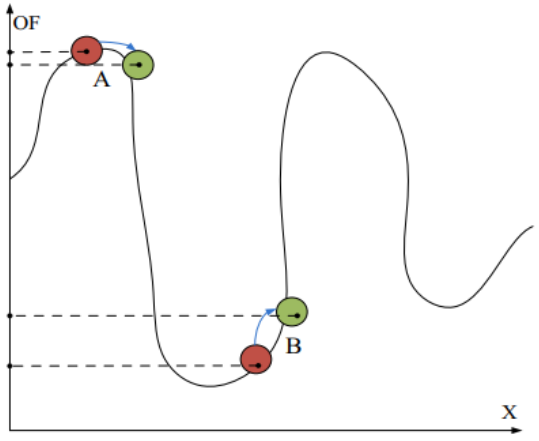
Da bismo pokrenuli algoritam, generiramo početnu populaciju veličine N_{pop}. Bira se skup najboljih mužjaka (jelena), a ostali su ženke (srne)$N_{srn} = N_{pop} - N_{muš}$.

4.4.5 Rika jelena

U ovoj fazi, jelen pokušava povećati svoju dobrotu rikanjem. Stoga, kao što se događa u prirodi, proces rikanja može biti uspješan ili neuspješan. Što se tiče prostora rješenja, nalazimo jelene susjede, a ako su ciljne funkcije susjeda bolje od drugog jelena, zamjenjujemo ih prethodnim. Dopušta se svakom jelenu da promjeni svoju poziciju. Za ažuriranje poziciju mužjaka koristi se sljedeća jednadžba:

$$muž_{novi} = \begin{cases} muž_{star} + a_1 \times \left((UB - LB)a_2\right) + LB), & ako\ a_3 \geq 0.5 \\ muž_{star} - a_1 \times \left((UB - LB)a_2\right) + LB), & ako\ a_3 < 0.5 \end{cases} \quad (3)$$

Da bi se stvorilo prihvatljivo rješenje u susjedstvu, UB i LB ograničavaju prostor za pretraživanje. Oni predstavljaju visoke granice (eng. upper bound) i niske granice (eng. lower bound) za prostore pretraživanja i rasponi postavljaju se nasumično. a_1, a_2 i a_3 prikazuju randomizaciju rike od 0 do 1. Da bi se uspostavila veza između ove jednadžbe i procesa rikanja u prirodi, mora se postaviti da kad jelen riče on pokušava proširiti svoj teritorij. Stoga se kreće nasumično.



Slika 21. Proces rike [61]

Slika iznad prikazuje proces rike jelena. Točka A i točka B označavaju riku. U slučaju točke A nova pozicija jelena je prihvaćena zbog dobrote/kvalitete rike (eng. objective fitness, OF) jer je to rješenje bolje od prethodnog. U slučaju točke B rješenje se ne može prihvatiti.

4.4.5.1 Podjela između običnog jelena i zapovjednika

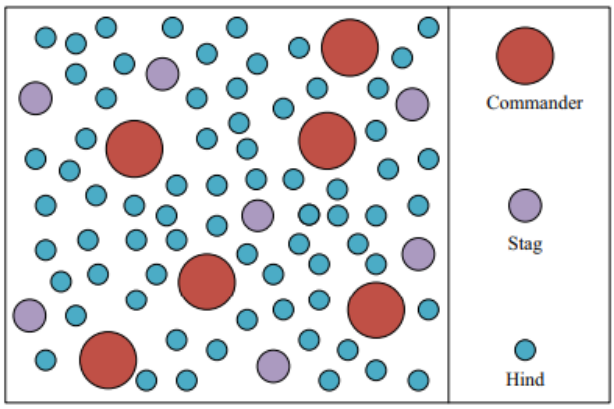
U prirodi postoji mnogo razlika između jelena. Neki su moćniji, privlačniji ili uspješniji u širenju teritorija od drugih. Prema tome su podijeljeni u grupe. Broj zapovjednika je predstavljen kao:

$$N_{zap} = \text{round}\{\gamma . N_{muž}\} \quad (4)$$

gdje je N_{zap} broj muških jelena, koji zapovijedaju haremom. γ je početna vrijednost modela algoritma, njegov raspon ide od 0 do 1. Broj običnih jelena je izračunat prema:

$$N_{običnJ} = N_{muž} - N_{zap} \quad (5)$$

gdje je $N_{običnJ}$ broj običnih jelena s obzirom na populaciju jelena. Slika 22. prikazuje populaciju jelena.



Slika 22. Populacija jelena [61]

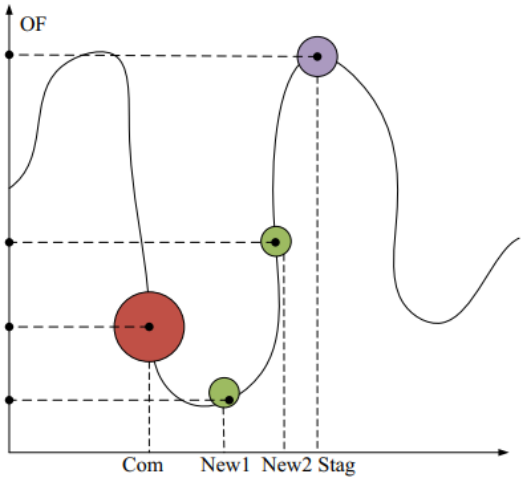
4.4.5.2 Borba između zapovjednika i običnog jelena

Dopušta se svakom zapovjedniku da se nasumično bori protiv jelena. Za prostor rješenja dopušta se da zapovjednici i suparnik približe jednom drugome. Zbog toga se dobivaju dva nova rješenja i zamjenjuje zapovjednika ako postoji bolje rješenje. Što se tiče procesa borbe, dane su dvije matematičke formule:

$$Nov_1 = \frac{(Zap+običnJ)}{2} + b_1 \times \left((UB - LB)b_2\right) + LB) \quad (6)$$

$$Nov_2 = \frac{(Zap+običnJ)}{2} - b_1 \times \left((UB - LB)b_2\right) + LB) \quad (7)$$

gdje su Nov_1 i Nov_2 dva nova rješenja nakon borbe. b_1 i b_2 predstavljaju nasumičnost borbe, njegove vrijednosti stoje između 0 i 1. Razmatrajući četiri rješenja, Zap, $običnJ$, Nov_1, Nov_2, s obzirom na njihovu dobrotu borbe (OF) najbolji ce biti odabran. Da bi se pokazala veza između navedenih formula i prirode jelena, mora se imati na umu da se u borbi zapovjednik i običan jelen približavaju jedni drugima. Jedan od njih će pobijediti u ovom nadmetanju, a drugi će izgubiti. Na taj se način generiraju dva nova rješenja.



Slika 23. Proces borbe [61]

Slika prikazuje primjer gdje je Nov_1 najbolje rješenje tako da po postupku Nov_1 je novi zapovijednik.

4.4.5.3 Stvaranje harema

Harem je skup srna skupljen od strane zapovjednika. Količina srna u haremu ovisi o snazi zapovjednika. Da bi se haremi formirali, srne se dijele proporcionalno između zapovjednika:

$$V_n = v_n \max_i\{v_i\} \tag{8}$$

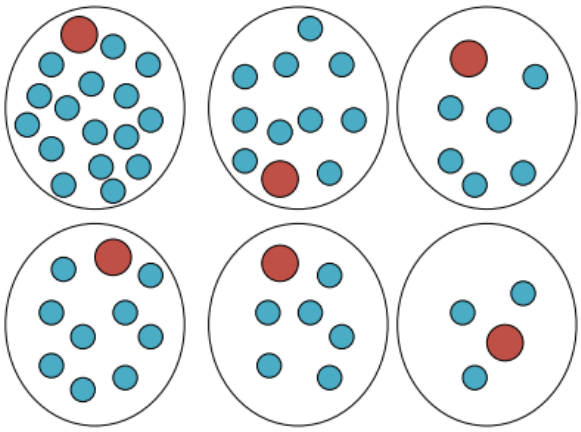
gdje v_n označava snagu n-tog zapovjednika (njegova dobrota, OF) dok V_n je normirana vrijednost. Da bi se izračunala normirana snaga zapovjednika, dana je sljedeća jednadžba:

$$P_n = \left| \frac{V_n}{\sum_{i=1}^{N_{zap}} V_1} \right| \tag{9}$$

S druge točke gledišta, normirana moć zapovjednika dio je značajki koje bi trebao posjedovati zapovjednik. Broj srna u haremu bio bi sljedeći:

$$N.\text{harem}_n = \text{round}\{P_n.N_{srn}\} \tag{10}$$

gdje $N.\text{harem}_n$ je broj srna u n-tom haremu, N_{srn} je broj srna. Podjela srne za svakog zapovjednika, bira se nasumično $N.\text{harem}_n$ od srna. Te će srne zajedno sa zapovjednikom osnovati harem. Što zapovjednik ima veću dobrotu/kvalitetu snage (OF) to će imati više srna.



Slika 24. Prikaz harema [61]

4.4.5.4 Parenje zapovjednika u svom haremu

Kao i druge vrste u prirodi, jeleni se međusobno pare. Ovu radnju izvodi zapovjednik, a određen postotak srna u njegovom haremu postat će roditelj.

$$N.\text{harem}_n^{par} = \text{round}\{\alpha.N.\text{harem}_n\} \tag{11}$$

gdje $N.\text{harem}_n^{par}$ je broj srna u n-tom haremu koje se pare sa zapovjednikom. Rješenje prostora prikazano je nasumično postavljanjem $N.\text{harem}_n^{par}$ od $N.\text{harem}_n$. α je početna vrijednost postavljena u modelu RDA, a raspon je od 0 do 1. Postupak parenja je formuliran na sljedeći način:

$$ptmk = \frac{(Zap+Srn)}{2} + (UB - LB)c \tag{12}$$

Zap i Srn su zapovijednik i srna, UB i LB su gornje i doljne granice, $ptmk$ je novo riješenje. Dok se c generira nasumično pomoću uniformne distribucijske funkcije gdje raspon je 0 do 1.

4.4.5.5 Parenje zapovjednika u drugim haremima

Nasumično se bira harem (zvan k) i dopušta se zapovjedniku da se pari sa β postotak srna u tom haremu. Zapovjednik je zapravo otišao u drugi Harem kako bi proširio svoj teritorij. β je početna vrijednost postavljena u modelu, raspon je od 0 do 1. Broj srna u tom haremu koje se pare sa zapovjednikom je :

$$N.\text{harem}_k^{par} = \text{round}\{\beta.N.\text{harem}_k\} \tag{13}$$

Gdje je $N.\text{harem}_k^{par}$ broj srna u k haremu koje se pare sa zapovjednikom.

4.4.5.6 Parenje običnog jelena s najbližom srnom

U ovoj fazi, običan jelen se pari sa najbližom srnom. Ova srna može biti njegova omiljena srna među svim srnama, isključujući haremska područja. Za pronalazak najbliže srne, udaljenost između običnog jelena i srne u J-dimenziji prostora se računa prema ovoj jednadžbi:

$$d_i = \left(\sum_{j\in J}\left(obič nJ_j - srn_j^i\right)^2\right)^{1/2} \quad (14)$$

gdje je d_i duljina između i-tog jelena i i-te srne. Prema tome, minimalna vrijednost u ovoj matrici predstavlja odabranu srnu. Nakon odabira najbliže srne počinje proces parenja. Proces parenja je isti kao kod parenja zapovjednika i srna (umjesto zapovjednika u formuli stavi se običan jelen).

4.4.5.7 Biranje sljedeće generacije

Za odabir sljedeće generacije primijenjene su dvije različite strategije. U prvoj, zadržavaju se svi jeleni, zapovjednici i obični jeleni (tj. postotak najboljih rješenja od svih rješenja). Druga strategija odnosi se na prisjećanje sljedeće generacije, biraju se srne između svih srna i potomaka dobivenih postupkom parenja, ovisno o kondiciji dobrote, koristi se mehanizam turnira [66] ili ruleta [67].

4.4.6 Rezultat algoritma s obzirom na kontrolne funkcije

Prvo, korišteno je 12 standardnih matematičkih kontrolnih funkcija. Sve one predstavljaju probleme minimiziranja. Optimalna vrijednost za sve njih je također nula. Glavni razlog odabira ovih kontrolnih funkcija je taj što svaka od njih ima skup posebnih karakteristika koje olakšavaju procjenu metaheuristike. Prvih 7 je unimodalno dok je drugih 5 mutimodalno. Ne samo da se predloženi RDA uspoređuje s nizom poznatih metaheuristika (GA, PSO), već se u tom pogledu razmatra i broj aktualnih algoritama optimizacije kao što su *imperialist competitive algorihm* (ICA) i algoritam krijesnica (eng. firefly algorithm) [68].

Početak
Input: Inizializacija populaciju jelena i srna
Output: Najbolje riješenje X^*
Izračunati dobortu i podijeliti u grupe jelena ($N_{muš}$) i srna (N_{srn})
X^* = najbolje riješenje
T1 = vrijeme.
dok (t < maksimalno vrijeme simulacije)
za svaki jelen
Rika jelena, jednadžba (3)
Ažuriranje pozicije ako je bolja od prijašnje
završi za
Sortiranje jelena na zapovjednike i na obične jelene sa jednadžbama (4)-(5)
za svakog zapovjednika
Borba između zapovjednika i običnog jelena, jednadžbe (6)-(7)
Ažuriranje pozicije zapovjednika i običnog jelena
završi za
Stvaranje harema, jedn. (8)-(10)
za svakog zapovjednika
jedn. (11)
Parenje između zapovjednika i nasumično određene srne u haremu,. jedn.(12)
Izabrati nasumično harem k
jedn. (13)
Parenje između zapovjednika i odabrane srne u k haremu, jedn. (12)
završi za
za svaki običan jelen
Izračunati udaljenost običnog jelena i srne, odabrati najbližu srnu, jedn. (14)
Parenje običnog jelena sa tom srnom, jedn. (12)
završi za
Izabrati iduću generaciju sa mehanizmom turnira ili rouleta.
Zaustaviti vrijeme
završi dok
vrati X^*
Kraj

Slika 25. Pseudokod za RDA. [61]

Algoritmi se izvršavaju 30 puta. Uzima se najbolja, najgora i srednja vrijednost. RDA ima najbolju konvergenciju u većini prvih kontrolnih funkcija, osim 1., 5. i 8. funkcije gdje drugi algoritmi dominiraju. Općenito, svi statistički rezultati RDA ukazuju na to da se u većini testnih zadataka pokazao pouzdan. Kao što je ranije spomenuto, algoritmi za optimizaciju sastoje se od dvije faze: istraživanja i eksploatacije. Faza istraživanja istražuje moguća područja u početnim iteracijama, a eksploatacija je rad pronalaženja dobrog rješenja u posljednjim iteracijama. Ova analiza pokazuje da su obje faze aktivirane u predloženom algoritmu ukoliko

ima više vremena za pronalaženje prostora rješenja. Na slici 25. nalazi se pseudokod za algoritam inspirian jelenima.

4.5 OPTIMIZACIJA TEMELJENA NA SUROM ORLU

Autor Abdolkarim Mohammadi-Balani predlaže metaheuristiku nadahnutu prirodom za rješavanje problema globalne optimizacije tzv. Optimizacija temeljena na surom orlu (eng. Golden eagle optimizer, skraćeno GEO) [69]. Glavni izvor inspiracije za GEO je inteligencija surog orla u podešavanju brzine u različitim fazama njihove spiralne putanje za lov.

4.5.1 Generalno

GEO se temelji na intelektualnim prilagodbama u napadu i načinu letenja koje suri orlovi izvode dok traže plijen i love. Pokazuju veću pozornost kretanju i pronalaženju plijena u početnim fazama lova i veću pozornost napadu u završnim fazama. Suri orao prilagođava ove dvije komponente tako da u najkraćem mogućem roku uhvati najbolji mogući plijen u određenoj regiji. Ovo se ponašanje matematički modelira kako bi se izdvojilo istraživanje i eksploatacija za metodu globalne optimizacije.

4.5.2 Inspiracija

Suri orao (kao što prikazan na donjoj slici) poznat kao *Aquila chrysaetos,* pripada obitelji *Accipitridae* koja obuhvaća različite vrste grabežljivih ptica poput sokola, orla i jastrebova. S izuzetnim vidom, velikom brzinom i moćnim kandžama, zlatni orlovi su profesionalni lovci koji mogu uhvatiti plijen različitih veličina, od insekata do sisavaca srednje veličine. Ova ptica može letjeti brzinom do 190 km / h. Za razliku od drugih vrsta orlova, može se naći na cijeloj sjevernoj hemisferi Zemlje [70].

Slika 26. Suri orao [70]

Jedinstvena značajka leta i lova surog orla je ta što se odvija spiralnom putanjom, što znači da je plijen većinu vremena s jedne strane orla. To im omogućuje da paze na predviđeni plijen, kao i na obližnje stijene i vegetaciju kako bi pronašli odgovarajući kut napada. Istodobno istražuju i druge regije kako bi pronašli bolju hranu [71].

U svakom trenutku leta, ponašanje surog orla određuju dva čimbenika: sklonost napadu i pozornost letenju. Suri orlovi znaju da ako napadnu na brzinu, mogu uhvatiti mali plijen koji čak ne nadoknađuje energiju utrošenu u lovu. S druge strane, ako se uključe u beskrajnu potragu za većim plijenom, možda će im ponestati energije i neće ništa uhvatiti. Suri orao inteligentno stvara ravnotežu između ove dvije želje kako bi ugrabio najbolji plijen u razumnom vremenu i utrošio razumnu količinu energije. Oni se glatko prebacuju s niskog napada-velike brzine leta na visoki napad-niske brzine leta. Svaki suri orao započinje lov, leteći na velikoj nadmorskoj visini unutar svog kraljevstva u velikim krugovima i tražeći plijen. Jednom kada primijeti plijen, počinje se kretati po obodu hipotetskog kruga usredotočenog na plijen. Suri orao pamti mjesto plijena, ali nastavlja kružiti iznad njega. Orao postupno smanjuje visinu i istovremeno se približava plijenu, čineći polumjer hipotetičkog kruga oko plijena sve manjim i manjim. Istodobno, istražuje i obližnje regije u potrazi za boljim alternativama. Ponekad suri orao dijeli mjesto najboljeg plijena koji je do sada pronašao s drugim orlovima. Ako orao ne primijeti bolje mjesto / plijen za lov, nastavlja kružiti oko trenutnog mjesta u manjim krugovima i na kraju napada plijen. Inače, ako orao pronađe bolju alternativu, napravi novi krug oko novog plijena i zaboravi na prethodni. Značajno je da se konačni napadi izvode u ravnoj liniji.

Dakle, glavne karakteristike postupka lova surog orla mogu se sažeti kako slijedi:

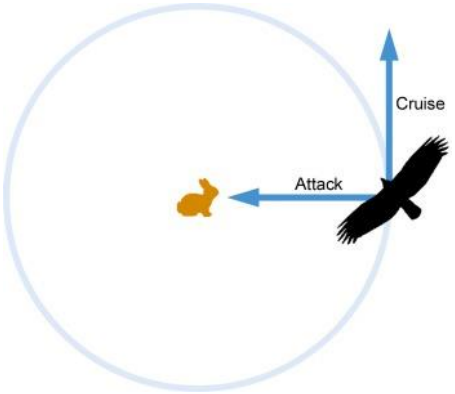
- Slijedi spiralnu putanju za pretraživanje i izravnu putanju za napad,
- Pokazuje veću pozornost letu u početnim fazama lova i glatko prelazi na veću pozornost napadu u završnim fazama,
- Zadržava mogućnost leta i napada u svakom trenutku leta,
- Traži informacije drugih orlova o plijenu.

Letenje, napad i razumna ravnoteža koju suri orao stvara su prirodna manifestacija istraživanja, eksploatacije i prelaska s prvog na drugo. To otvara put razvoju ovog metaheurističkog algoritma.

4.5.3 Algoritam

4.5.3.1 Spiralno kretanje suri orlova

GEO se temelji na spiralnom kretanju surih orlova. Kao što je ranije spomenuto, svaki suri orao pamti najbolje mjesto koje je do sada posjetio. Orao istodobno osjeća privlačnost da napadne plijen i leti u potrazi za boljom hranom. Moguće je vizualizirati vektore napada i leta u prostoru na donjoj slici.



Slika 27. Spiralno kretanje surog orla [69]

U svakoj iteraciji, svaki suri orao i nasumično bira plijen drugog surog orla f i kruži oko najboljeg mjesta koje je suri orao f do sada posjetio. Suri orao i također može odlučiti kružiti oko vlastitog plijena, $VelPop$ je veličina populacije surih orlova.

$$f \in \{1,2,\ldots,VelPop\}$$

4.5.3.2 *Odabir plijena*

U svakoj iteraciji svaki suri orao mora odabrati plijen za obavljanje operacija leta i napada. Plijen je modeliran kao najbolje rješenje koje je do tog momenta jato surih orlova uspjelo pronaći. Svaki suri orao može se sjetiti najboljeg rješenja koje je do sada pronašao. U svakoj iteraciji, svaki agent za pretraživanje odabire ciljni plijen iz memorije cijelog jata. Vektori leta i napada surog orla se izračunavaju u odnosu na odabrani plijen. Ako je novi položaj (izračunat pomoću vektora napada i leta) bolji od prethodnog položaja u memoriji, tada se memorija ažurira. Odabir plijena igra važnu ulogu u GEO-u. Odabir se može dogoditi na jednostavan način, pri čemu svaki suri orao odabire plijen samo iz vlastite memorije. Da bi se pomoglo surim orlovima da bolje istraže teritorij, postavlja se da svaki suri orao nasumično odabire svoj plijen u trenutnoj iteraciji iz memorije bilo kojeg drugog člana jata. Značajno je da odabrani plijen nije nužno najbliži ili najudaljeniji plijen. U ovoj shemi, svaki plijen u memoriji dodjeljuje se jednom i samo jednom orlu.

4.5.3.3 *Napad (eksploatacija)*

Napad se može simulirati vektorom koji započinje trenutnim položajem orla i završava položajem plijena. Vektor napada za surog orla može se izračunati pomoću:

$$\vec{A}_i = \vec{X}_f^* - \vec{X}_i \tag{1}$$

gdje $\vec{A}_i$ je vektor napada orla, $\vec{X}_f^*$ je najbolja pozicija pljena koje je orao f do sada posjetio, $\vec{X}_i$ je trenutna pozicija orla i, budući da vektor napada usmjerava populaciju surih orla na najposjećenija mjesta, on naglašava fazu eksploatacije u GEO.

4.5.3.4 *Let (istraživanje)*

Vektor leta izračunava se na temelju vektora napada. Vektor leta je vektor tangenta na krug i okomit na vektor napada. Let se također može smatrati linearnom brzinom surog orla u odnosu na plijen. Vektor leta u n dimenzijama nalazi se unutar tangente hiper-ravnine na krug; dakle, da bi se izračunalo vektor leta, prvo se mora izračunati jednadžbu tangente hiper-ravnine. Jednadžba prikazuje skalarni oblik jednadžbe hiper-ravnine u n dimenzionalnom prostoru:

$$h_1 x_1 + h_2 x_2 + \ldots + h_n x_n = d \Rightarrow \sum_{j=1}^{n} h_j x_j = d \tag{2}$$

gdje je $\vec{H} = [h_1, h_2, \ldots, h_n]$ normalni vektor, $\vec{X} = [x_1, x_2, \ldots, x_n]$ je vektor varijabli, $\vec{P} = [p_1, p_2, \ldots, p_n]$ je proizvoljna točka u hiper-ravnini, i $d = \vec{H} \cdot \vec{P} = \sum_{j=1}^{n} h_j p_j$. Ako se uzme u obzir $\vec{X}_i$ (pozicija orla i) kao proizvoljna točka u hiper-ravnini i da je $\vec{A}_i$ (vektor napada) normala od hiper-ravnine, može se pokazati jednadžba gdje $\vec{C}_i^t$ (je vektor leta za orla i u iteraciji t) pripada hiper-ravini:

$$\sum_{j=1}^{n} a_j x_j = \sum_{j=1}^{n} a_j^t x_j^* \tag{3}$$

gdje je $\vec{A}_i = [a_1, a_2, \ldots, a_n]$, vekoror napada, $X = [x_1, x_2, \ldots, x_n]$ je vektor odluke/dizajna i $X^* = [x_1^*, x_2^*, \ldots, x_n^*]$ je pozicija odabranog plijena.

Sada kada je izračunata hiper-ravnina za let u iteraciji t za orla i, mora se pronaći vektor leta za tog orla u hiper-ravnini. Suri orao može odabrati bilo koje odredište u hiper-ravnini. Da bi se pronašlo nasumični vektor na hiper-ravnini leta, prvo se mora odrediti točka C koja ima nasumično odredište u hiper-ravnini koja je postavljena (trenutna pozicija surog orla i). Početna točka vektora leta je trenutna pozicija orla i. S obzirom da su hiper-ravnine jednu dimenziju manje od prostorne dimenzije, postavlja se da postoji $n - 1$ dimenzija gdje početna točka može biti izabrana po želji. Zadnja dimenzija mora biti postavljena da zadovoljava jednadžbu hiper-ravnine, znači da ima $n - 1$ slobodnih varijabla i jedna fiksna varijabla. Koristi se sljedeći postupak da bi se pronašla nasumična n-dimenzionalna točka C postavljena na hiper-ravnini leta za orla i.

Najprije se nasumično odabere jedna fiksna varijabla od n varijabli. Ta varijabla se zove k (naravno ne može se koristiti kao fiksna varijabla kada je vektor napada 0, zbog toga što varijabla može biti bilo koja nasumična kombinacija od ostalih $n - 1$ varijabla).

Drugo, dodjeljuju se nasumične vrijednosti svim varijablama osim k varijabli jer je fiksna. Za pronalazak vrijednosti fiksne varijable koristi se sljedeća jednadžba:

$$c_k = \frac{d - \sum_{j, j \neq k} a_j}{a_k} \tag{4}$$

gdje c_k je k-ti elemnt od krajne točke C. a_j je j element od vektora napda $\vec{A}_i$, a_k je k element od vektora napada $\vec{A}_i$, dok je k indeks fikse varijable. Nasumična točka na hiper-ravnini je pronađena. Za reprezentaciju odredišta na hiper-ravnini leta koristi se sljedeća jednadžba:

$$\vec{C}_i = \left(c_1 = \text{random}, c_2 = \text{random}, \ldots, c_k = \frac{d - \sum_{j, j \neq k} a_j}{a_k}, \ldots, c_n = \text{random}\right) \quad (5)$$

Nakon što je točka odredišta određena, moguće je izračunati vektor leta za surog orla i u iteraciji t. Elementi rezultirajuće odredišne točke su slučajni brojevi od nule do jedan. Važno za znati e da vektor leta privlači populaciju surih orlova na područja različita od onih navedenih u memoriji, to naglašava fazu istraživanja u GEO.

4.5.3.5 Pomak u nove pozicije

Pomicanje surog orla sastoji se od vektora napada. Vektor koraka za orla i u iteraciji t definiran je s jednadžbom:

$$\Delta x_i = \vec{r}_1 p_a \frac{\vec{A}_i}{\left|\left|\vec{A}_i\right|\right|} + \vec{r}_2 p_c \frac{\vec{C}_i}{\left|\left|\vec{C}_i\right|\right|} \quad (6)$$

gdje je p_a^t koeficijent napada u t iteraciji a p_c^t je koeficijent leta u t iteraciji, ti koeficijenti određuju kako na sure orle utječu napad i let. p_a^t koeficijent napada i p_c^t koeficijent leta kontroliraju kako na vektor koraka utječu vektori napada i leta. $\vec{r}_1$ i $\vec{r}_2$ su nasumični vekotori od 0 do 1. $\left|\left|\vec{A}_i\right|\right|$ i $\left|\left|\vec{C}_i\right|\right|$ su euklidska norma za vektore napada i leta i računaju se prema jednadžbi:

$$\left|\left|\vec{A}_i\right|\right| = \sqrt{\sum_{j=1}^{n} a_j^2}, \qquad \left|\left|\vec{C}_i\right|\right| = \sqrt{\sum_{j=1}^{n} c_j^2} \quad (7)$$

Pozicija surih orlova u $t + 1$ iteraciji se računa tako da se doda Vektor koraka na poziciji orla od iteracije t.

$$x^{t+1} = x^t + \Delta x_i^t \quad (8)$$

Ako je pozicija orla i bolja od pozicija u sjećanju, memorija se ažurira sa novom pozicijom. Ako ne, memorija ostaje ista, ali orao će ostati u novoj poziciji.

4.5.3.6 Tranzicija od istraživanja u eksploataciji

Suri orlovi pokazuju veću pozornost letu u početnim fazama lova što odgovara istraživanju i prelaze na veću pozornost napadu u završnim fazama što odgovara eksploataciji. GEO koristi p_a i p_c za promjenu istraživanja i eksploatacije. Algoritam počne s niskim p_a i visokim p_c, kako se iteracije povećavaju tako se p_a povečava a p_c se smanjuje. Korisnik determinira vrijednosti.

$$\begin{cases} p_a = p_a^0 + \frac{t}{T}|p_a^T - p_a^0| \\ p_c = p_c^0 + \frac{t}{T}|p_c^T - p_a^0| \end{cases} \quad (9)$$

gdje je t trenutna iteracija, T je maksimum broj iteracija, p_a^0 i p_a^T su početna i krajna vrijenost od pozornosti napada, dok p_c^0 i p_c^T su početna i krajna vrijednost za pozornost leta.

Početak
Inicijalizacija populacije surih orlova
Procijeniti dobrotu funkcije
Inicijalizirati populaciju iz memorije
Inicijalizirat p_a i p_c
  za svaku iteraciju t
    Ažuriraj p_a i p_c s jednadžbom (9)
    za svakog surog orla i
      Nasumično odabrati pljen iz memorije populacije
      Izračunati vektor napada $\vec{A}$, jedn. (1)
      ako dužina vektora napada $\neq 0$
        Izračunat vektor leta $\vec{C}$, jedn. (2)-(5)
        Iztačunati vektor koraka Δx, jedn (6)-(8)
        Ažurirati poziciju, jedn. (8)
        Procijeniti dobrotu funkcije za novu poziciju
        ako dobrota pozicije je bolja nego dobrota pozicije u memoriji orla i
          Promjeniti novu poziciju sa prijašnjom poziciji u memoriji orla i
        završi ako
      završi ako
    završi za
  završi za
Kraj

Slika 28. Pseudokod GEO algoritma [69].

4.5.4 Rezultati

Da bi se numerički dokazalo spomenute teorijske tvrdnje, provodi se širok spektar eksperimenata. Kontrolne funkcije su podijeljene u 3 klase. Unimodalne, multimodalne i kompozicijske funkcije, od toga su 7 unimodalne, 15 su multimodalne i 10 su kompozicijske funkcije. Prema prikazanim rezultatima vidi se da GEO ima mogućnosti šire pretrage, ali daje veći naglasak na pretragu obećavajućih područja. Može se vidjeti da se u unimodalnim funkcijama krivulja konvergencije kontinuirano poboljšava. Međutim, to možda nije slučaj za multimodalne i kompozitne funkcije, gdje je GEO izložen mnogim lokalnim minimumima i možda neće posjetiti najbolje pozicije za neke iteracije. Iako su kvalitativne metrike potvrdile mogućnosti istraživanja i eksploatacije GEO-a, one ne mogu u potpunosti prikazati koliko se dobro može riješiti problem optimizacije. Učinjeno je 30 zasebnih pokretanja da bi se što bolje

prikazali statistički pokazatelji kvalitete za GEO. Aritmetička sredina pokazuje kako GEO radi u prosjeku, ali važno za spomenuti da je sa standardnom devijacijom, algoritam jako stabilan. Iz rezultata se može vidjeti da je GEO superiorniji od ostalih algoritama u polovici unimodalnih funkcija i da daje konkurentne rezultate u ostalim unimodalnim funkcijama. Za multimodalne funkcije GEO nadmašuje druge algoritme u 13 od 16 kontrolnih funkcija. Dok koristeći srednju vrijednost dobrote GEO nadmašuje 8 od 10 kompozicijskih funkcija kod testiranja svih algoritama. Na slici 28. nalazi se pseudokod za optimizaciju temeljenu na surom orlu.

4.6 OPTIMIZACIJSKI ALGORITAM PRETRAZIVANJA TEMELJEN NA GUŠTERU

Ovaj rad napisao je Neetesh Kumar i pokušava modelirati dinamičko ponašanje guštera *Agame* u potrazi za hranom i njihov učinkovit način hvatanja plijena u matematičkom modelu koji se naziva optimizacijski algoritam pretraživanja temeljen na gušteru (eng. Artificial lizard search optimization, skraćeno ALSO) [72].

4.6.1 Generalno

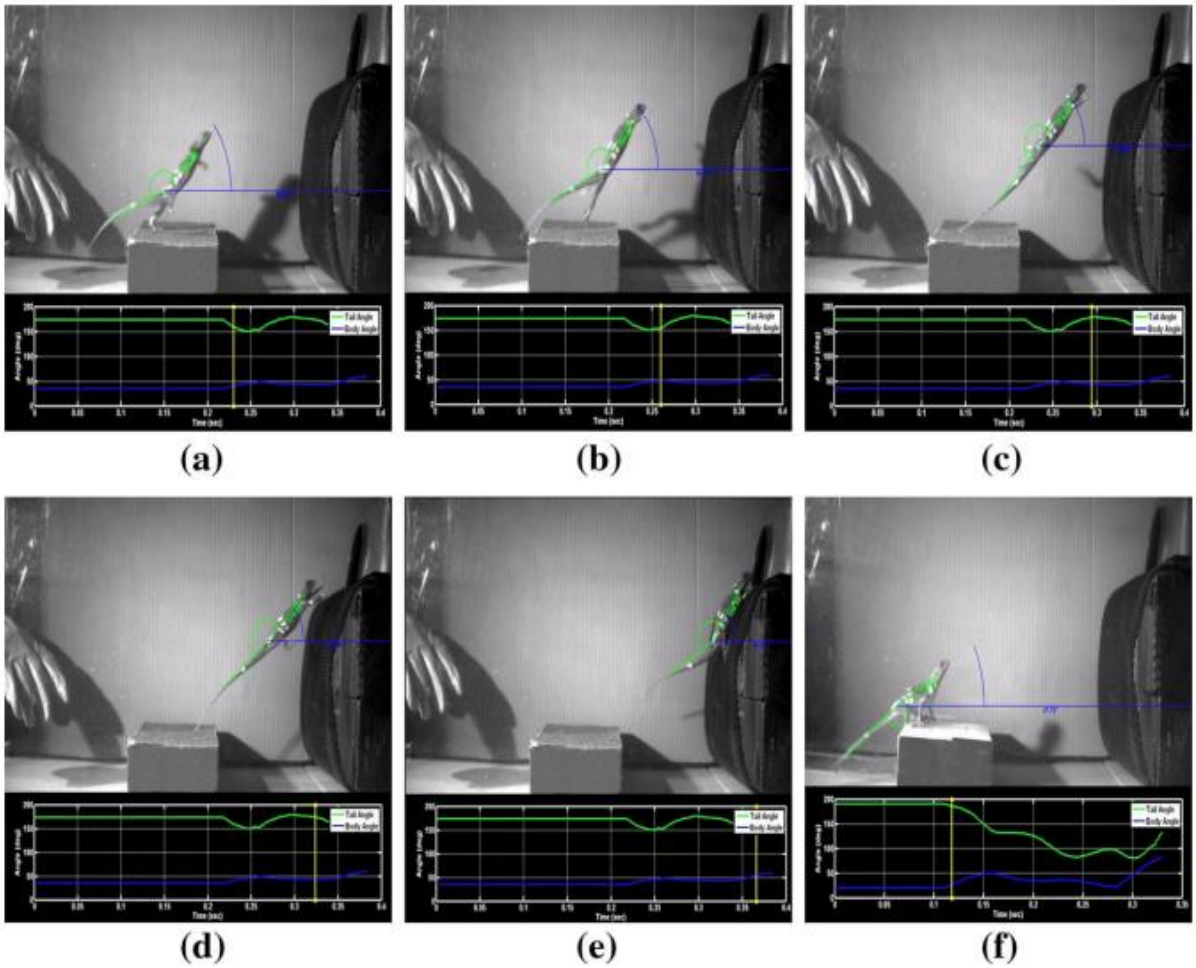
Ovaj rad predlaže novi i učinkovit algoritam inspiriran prirodom, nazvan ALSO za ograničene probleme optimizacije temeljene na aplikacijama. Ideja se temelji na nedavnoj studiji u kojoj su istraživači izvijestili da gušteri kontroliraju kretanje repova kako bi preusmjerili kutni moment iz tijela prema repovima, stabilizirajući položaj tijela u sagitalnoj ravnini. Predloženi algoritam također se temelji na tome kako crveno-glavati gušteri *Agama* hvataju svoj plijen. Oponaša dinamično ponašanje guštera *Agame* u potrazi za hranom i njihov učinkovit način dobro organiziranog hvatanja plijena. ALSO pokušava odmjereno kontrolirati zamah repa guštera kako bi preusmjerio kutni moment iz njihovih tijela prema repovima, stabilizirajući položaj tijela u sagitalnoj ravnini.

4.6.2 3.6.2. Inspiracija

Ideja ovog algoritma proizlazi iz rada koji je 2012 godine napisao Libby [73]. Provedeno je niz eksperimenata u laboratoriju kako bi analizirali skakanje crvenoglavnih guštera *Agame* prilikom hvatanja plijena tijekom procesa lova. Da bi izveli niz eksperimenata, koristili su žive *Agame* guštere u svom laboratoriju i zabilježili njihove pokrete u obliku videozapisa kao što je pikazano na Slici 29.

Predloženi ALSO uzima u obzir crvenoglave *Agame* guštere koji su spremni na svojim slučajnim položajima (u višedimenzionalnom prostoru za pretraživanje) kako bi pronašli hranu. Da bi dobili hranu, gušteri love prehrambene resurse (insekte) skačući sa svog mjesta na plijen. Gušteri mijenjaju svoj kutni položaj i brzinu tijekom skoka (tj. let). Kutna brzina određena je zakretnim momentom za ispitivanje plijena u prostoru za pretraživanje. Mahali bi repovima dok su skakali, što je odgovaralo kontroli nagiba tijela. Promjena kuta repa događa se u odnosu na tijelo tijekom faze leta. Uz to, gušteri bi mahali repovima prema gore kako bi nadoknadili poremećaje usmjerene prema dolje i prema dolje kao odgovor na poremećaje usmjerene prema gore. Rotirajući repove u odnosu na tijelo u sagitalnoj ravnini, gušteri prenose kutni moment od tijela do repa, smanjujući tako kutnu brzinu tijela i učinak poremećaja na rotaciju tijela, kako je pokazano da donjoj slici. Nakon toga, gušteri mijenjaju svoj stvarni položaj podešavanjem kutnog položaja tijela i repa odgovarajućim okretnim momentom i brzinom (tijekom skoka) na kontrolirani i uravnoteženi način kako bi došli do svog plijena.

Da bi se iskoristila ta inteligencija, definira se iterativni postupak, a nekoliko guštera raspoređeno je na slučajne položaje u višedimenzionalnom prostoru pretraživanja, gdje ti gušteri skaču sa slučajnim kutovima tijela, repa i okretnim momentom. Nakon ovog skoka, gušteri postižu određeni položaj (rješenje) pomoću funkcije dobrote za mjerenje kvalitete rješenja. S time se globalno najbolje (g_{bolji}) i lokalno najbolje (l_{bolji}) rješenje ažuriraju. Najbolja pozicija u bilo kojoj iteraciji je l_{bolji} dok je najbolja pozicija do zadnje iteracije g_{bolji}. U sljedećoj iteraciji stvarni položaj guštera ažurira se ažuriranjem kutne brzine, položaja i zakretnog momenta. Kvaliteta rješenja mjere se u svakoj iteraciji, l_{bolji} i g_{bolji} se ažuriraju. Ovaj se postupak ponavlja sve dok se ne postigne kriterij završetka.



Slika 29. Različite pozicije tijela i repa Agame guštera [73]

4.6.3 Algoritam

ALSO počinje sa inicijalizacijom varijabli, tj. brojač iteracija k, $gbolji$ i $lbolji_i(k)$. Isto tako za dimenzionalni vektor pozicije $x_i(k)$ za i-ti gušter, $\theta_{ib}(k)$ i $\theta_{it}(k)$ za kut tijela i kut repa koje predstavljaju izvedenicu od kuta segmenta i inicijalizacija τ za okretni moment. Raspon kuta tijela ide od [-45°, 45°], kuta repa [-90°, -90°], a okretni moment [0, 1]. U svakoj iteraciji se dodjeljuje vrijednost dobrote svakom gušteru, te se procjenjuje preko funkcija dobrote koja određuje kvalitetu dobivenog rješenja.

4.6.3.1 Kodiranje

Slično drugim metodama optimizacije koje se temelje na populaciji, također se započinje slučajnim početnim položajem guštera koji skaču. Položaj guštera koji skače označen je dimenzionalnim vektorom u prostoru pretrage. Dakle, gušter može skočiti u hiperdimenzionalni / višedimenzionalni prostor za pretraživanje i može promijeniti svoj

položajni vektor. Položaj guštera koji skače (eng. leaping lizard, LL) u višedimenzionalnim prostoru, prikazan je u doljnoj jednadžbi:

$$LL = \begin{bmatrix} LL_{1,1} & LL_{1,2} & \cdots & LL_{1,d} \\ LL_{2,1} & LL_{2,2} & \cdots & LL_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ LL_{n,1} & LL_{n,2} & \cdots & LL_{n,d} \end{bmatrix} \quad (1)$$

gdje $LL_{i,j}$ označava j-ti dimenziju za i-ti gušter. Za uniformnu distirbuciju guštera u prostoru koristi se doljnja jednadžba za raspodijelu početne vrijednosti položaja svakog guštera u višedimenzionalnim prostoru pretraživanja. Položaj vektora za i-ti gušter za k-tu iteraciju i za j-te dimenzije se postavlja kao:

$$x_{i,j}(k) = LL_{min} + (LL_{max} - LL_{min})r \quad (2)$$

gdje su LL_{min} i LL_{max} varijable za određivanje minimalnih i maksimalnih granica prostora pretraživanja, r je nasumični broj od [0, 1].

4.6.3.2 Procjena dobrote

„Dobrota“ svakog vektora položaja koji odgovara svakom gušteru procjenjuje se pomoću varijable odluke (rješenje) i postavlja se definirana funkcija dobrote koja ovisi o problemu. Nakon provjere vrijednosti kvalitete/dobrote se pohranjuju u:

$$F = \begin{bmatrix} F_1([LL_{1,1} & LL_{1,2} & \cdots & LL_{1,d}]) \\ F_2([LL_{2,1} & LL_{2,2} & \cdots & LL_{2,d}]) \\ \vdots & \vdots & \vdots & \vdots \\ F_n([LL_{n,1} & LL_{n,2} & \cdots & LL_{n,d}]) \end{bmatrix} \quad (3)$$

Vrijednost dobrote položaja guštera koji skače odražava kvalitetu dobivenog rješenja. Veća vrijednost dobrote ukazuje na to da je gušter blizu ciljanog rješenja.

4.6.3.3 Ažuriranje pravila

- **Okretni moment (Torque τ):** igra važnu ulogu tijekom skoka guštera. U ovom se modelu koristi τ za ažuriranje izvedenice tijela i repa svakog guštera u iteracijama. Osim toga, τ je također važan za kontrolu brzine konvergencije algoritma tijekom ažuriranja položaja. Stoga je predložen učinkovit mehanizam za osvježavanje okretnog momenta koji se temelji na slučajnom ponašanju, ali s učinkovitom povezanošću s kvalitetnom odlukom u prethodnoj iteraciji. $\tau_i(k+1)$ (označuje okretni moment u sljedećoj iteraciji) ima raspon između 0 i q gdje je q :

$$q = \begin{cases} \tau_i(k), & \text{ako } x_i(k) = lbolji(k) \\ \frac{F(gbolji) - F(x_i)}{F(gbolji) - F(x_{gori})}, & inače \end{cases} \quad (4)$$

Osnovna ideja koja stoji iza ove formulacije je da se sljedeći τ generira na temelju dobrote rješenja $gbolji$. Ako je rješenje predaleko od $gbolji$, tada je generirani τ širi, ako je bliže najboljem rješenju onda je domet kraći. Ako gušter ima vrijednost $gbolji$ onda je vrijednost $\tau_i(k)$ ista kao prošla iteracija koja je najbolja za guštera. Pri tome se uspostavlja veza između $gbolji$ i drugih rješenja, gdje se τ pokušava što više približiti $gbolji$ riješenju.

- **Kut ($\boldsymbol{\theta_i(k)}$):** Da bi se ažuriralo $\theta_i(k)$ (vrijedi za kut tijela i tepa) koristi se ova formulacija:

$$\theta_i(k) = f\big(\theta_i(k)\big) + g(\theta_i(k)) \times \tau_i(k) \quad (5)$$

gdje $f\big(\theta_i(k)\big)$ i $g(\theta_i(k))$ označava kutnu položaj i kutnu brzinu za i-ti gušter u k iteraciji, definirani su kao:

$$f\big(\theta_i(k)\big) = \left[\theta_{ib}(k), \frac{{\theta_{ib}}^2(k) - b}{d - e}, \theta_{it}(k), \frac{-{\theta_{it}}^2(k) + c}{d - e}\right] \quad (6)$$

$$g\big(\theta_i(k)\big) = \left[0, -\frac{f}{d-e}, 0, \frac{g}{d-e}\right] \quad (7)$$

gdje

$$a = \frac{1}{2} l_{ib}^2 l_{it}^2 m_{ib}^2 m_{it}^2 \sin(2(\theta_{ib}(k) - \theta_{it}(k)))$$

$$b = \left(l_{it}^2 m_{ib} m_{it} + I_{it}(m_{ib} + m_{it})\right) l_{ib} l_{it} m_{ib} m_{it} {\theta_{it}}^2(k) \sin(\theta_{ib}(k) - \theta_{it}(k))$$

$$c = \left(l_{ib}^2 m_{ib} m_{it} + I_{ib}(m_{ib} + m_{it})\right) l_{ib} l_{it} m_{ib} m_{it} {\theta_{it}}^2(k) \sin(\theta_{ib}(k) - \theta_{it}(k))$$

$$d = (l_{ib}^2 l_{it}^2 m_{ib}^2 m_{it}^2 \cos(\theta_{ib}(k) - \theta_{it}(k)))^2 \quad (8)$$

$$e = \left(l_{ib}^2 m_{ib} m_{it} + I_{ib}(m_{ib} + m_{it})\right) (l_{it}^2 m_{ib} m_{it} + I_{ib}(m_{ib} + m_{it}))$$

$$f = \left(l_{it}^2 m_{ib} m_{it} + I_{it}(m_{ib} + m_{it})\right) - l_{ib} l_{it} m_{ib} m_{it} \cos(\theta_{ib}(k) - \theta_{it}(k))(m_{ib} + m_{it})$$

$$g = \left(l_{ib}^2 m_{ib} m_{it} + I_{ib}(m_{ib} + m_{it})\right) - l_{ib} l_{it} m_{ib} m_{it} \cos(\theta_{ib}(k) - \theta_{it}(k))(m_{ib} + m_{it})$$

$I_{ib} = m_{ib} l_{ib}^2$ i $I_{it} = m_{it} l_{it}^2$. Budući da su gušteri s velikim repom učinkovitiji, njihovo tijelo, duljina repa i masa mogu se uzeti prema uputama gdje vrijedi l_{ib} je $\frac{l}{3}$ a l_{it} je $\frac{2l}{3}$, zbog toga se mogu postaviti iduće vrijednosti:

$$a = \frac{1}{5000} m^4 l^4 \sin(2(\theta_{ib}(k) - \theta_{it}(k)))$$

$$b = \frac{19}{11250} m^4 l^4 {\theta_{it}}^2(k) \sin(\theta_{ib}(k) - \theta_{it}(k))$$

$$c = \frac{11}{5000} m^4 l^4 {\theta_{it}}^2(k) \sin(\theta_{ib}(k) - \theta_{it}(k))$$

$$d = \frac{1}{2500} m^4 l^4 \cos^2(\theta_{ib}(k) - \theta_{it}(k)) \quad (9)$$

$$e = \frac{209}{22500} m^4 l^4$$

$$f = \frac{1}{900} m^3 l^2 (56 - 9\cos(\theta_{ib}(k) - \theta_{it}(k)))$$

$$g = \frac{1}{100} m^3 l^2 (11 - 2\cos(\theta_{ib}(k) - \theta_{it}(k)))$$

Budući da gornje vrijednosti ovise samo o $\theta_{ib}(k)$ i $\theta_{it}(k)$ i uzejući u obzir da su masa i dužina guštera jedinični, onda može se definirati ovako:

$$a = \sin(2(\theta_{ib}(k) - \theta_{it}(k)))$$

$$b = {\theta_{it}}^2(k) \sin(\theta_{ib}(k) - \theta_{it}(k))$$

$$c = b$$

$$d = \cos^2(\theta_{ib}(k) - \theta_{it}(k)) \quad (10)$$

$$e = 1$$

$$\mathrm{f} = 56 - 9\cos(\theta_{ib}(k) - \theta_{it}(k))$$

$$\mathrm{g} = 11 - 2\cos(\theta_{ib}(k) - \theta_{it}(k))$$

- **Vektor položaja($\mathbf{x_i(k)}$):** Nakon procijene dobrote svake čestice, $lbolji$ od trenutne iteracije je ažuriran iz naboljeg položaja. I $gbolji$ riješenje je također ažuriran. Nadalje

s ažuriranjem kutnog položaja, brzine, okretnog momenta i izvedenice kutova tijela i repa, ažurira se vektor položaja za i-ti gušter u k iteraciji, kao što je prikazano u idućoj jednadžbi:

$$\mathrm{x_i(k+1)} = \mathrm{x_i(k)} + \tau_i(k+1) \times (0.3) \times \Delta\theta + c_1 \times rand \times \left(lbolji(k) - \mathrm{x_i(k)}\right) + c_2 \times rand \times (gbolji - \mathrm{x_i(k)}) \quad (11)$$

gdje $\Delta\theta$ je razlika između kuta tijela i repa u stupnjevima, za radijane je postavljeno:

$$\Delta\theta = (\theta_{ib}(k) - \theta_{it}(k)) \times \frac{\pi}{180} \quad (12)$$

$x_i^h(k)$ je pozicija guštera u k iteraciji. U formulaciji τ je okretni moment, c_1 i c_2 su konstante, $rand$ su uniformni borjevi između 0 i 1.

4.6.3.4 Kriteriji za završetak algoritma

Uvjet završetka definira ALSO iterativni postupak koji se treba zaustaviti. Predloženi su tri kriterija : 1. kada se dobije optimalno rješenje, 2. kada se postigne maksimalni broj iteracija i 3. konvergencija svih guštera na istu točku. Generalno može se postaviti bilo kakv kriterij za kraj algoritma.

4.6.4 Rezultat

Da bi se analizirala izvedba predloženog algoritma, provode se rigorozni eksperimenti s nekoliko klasičnih, kao i suvremenih kontrolih funkcija koje rješavaju probleme optimizacije. 32 kontrolne funkcije su standardne i klasične, te funkcije su unimodalne, multimodalne, kontinuirane, nekontinuirane, linearne, nelinearne, koveksne i nekonvekse.

Od ukupno 128 kvantitativnih ocjena (po 4, najbolje, najgore, prosječno i standardna devijacija za 32 značajke), predloženi model nadmašio je ostale sa 104 procjene, što je 81,25%. Za ostale funkcije, predloženi model dosegnuo je drugi rang. Značajno poboljšanje performansi u ALSO posljedica je njegovog jedinstvenog mehanizma za pronalaženje hrane, jer svaki gušter pokušava pogoditi metu podešavanjem kutova tijela i repa. Za podešavanje kutova nagiba trupa i repa koristi svoja najbolja lokalna i globalna rješenja za stvaranje učinkovitog okretnog momenta koji uvelike pomaže u podešavanju kutova nagiba trupa i repa u idućoj iteraciji. Stoga je rješenje koje je razvio ALSO vrlo učinkovito u usporedbi s drugim modernim rješenjima. Međutim, mehanizam stvaranja zakretnog momenta može se poboljšati korištenjem drugih mehanizama učenja. Na slici 30. nalazi se pseudokod za algoritam pretraživanja temeljen na gušteru.

Početak
$k = 0$
Inicijalizacija g_{bolji} i $l_{bolji}(0)$ kao dimenzionalne vektore
Nasumično inicijalizirat dimenzionalni vektor pozicije $x_i(k)$, kut tijela $\theta_{ib}(k)$, kut repa $\theta_{it}(k)$, i okretni moment τ
dok nemamo kondiciju završetka
za svakog guštera i
Procijeniti dobrotu funkcije $F(x_i(k)) = Fitness(x_i(k))$
završi za
Ažurirat najbolju lokalnu poziciju za k iteracija prema
$l_{bolji}(k) = x_i(k); \forall j \neq i$ i $F(x_i(k)) < F(x_j(k))$.
Ažurirat najbolju globalnu poziciju za k iteracija prema

$$g_{bolji} = \begin{cases} g_{bolji}, & \text{ako } F(g_{bolji}) < F(l_{bolji}(k)) \\ l_{bolji}(k), & \text{inače} \end{cases}.$$

za svaki gušter i
Nasumično stvaranije vrijednosti za $\tau_i(k+1)$ u rasponu [0 ... q], sa jedn. (4)
Razmotri se stanje di $\theta_i(k) = [\theta_{ib}(k)\dot{\theta}_{ib}(k)\theta_{it}(k)\dot{\theta}_{it}(k)]^T$ za jedn. (5)
Ažuriraj vektor pozicije prema $x_i(k+1) = x_i(k) + \tau_i(k+1) \times (0.3) \times \Delta\theta +$ $c_1 \times rand \times \left(l_{bolji}(k) - x_i(k)\right) + c_2 \times rand \times (g_{bolji} - x_i(k))$
završi za
$k = k + 1$
završi dok
Kraj

Slika 30. Pseudokod ALSO algoritma [72]

5. ZAKLJUČAK

Za rješavanje velike većine problema optimizacije u stvarnom svijetu, heuristika je i ostat će jedina opcija, bez obzira je li dizajnirana pomoću metaheurističkog razvojnog okvira ili ne.

Istraživači često hvale prirodu kao savršen primjer adaptivnog rješavanja problema pa stoga ne čudi da su algoritmi metaheurističke optimizacije nadahnuti prirodnim fenomenima postali toliko popularni. Te su metode razvijene kako bi oponašale bilo koju biološku ili fizičku manifestaciju koji se promatra u prirodi, s ciljem razvijanja snažnih alata koji se mogu primijeniti za rješavanje problema optimizacije.

Jedna od prednosti metaheurističkih algoritama je jednostavnost rada. Glavna prednost metaheuristike nadahnute prirodom u odnosu na tradicionalne metode optimizacije leži u njihovoj sposobnosti rješavanja širokog spektra problema bez obzira na njihovu strukturu i svojstva. Zahvaljujući ovoj prepoznatljivoj osobini, ove su tehnike postale popularan izbor za rješavanje složenih problema. Kao rezultat toga, ove su tehnike pronašle primjenu u gotovo svim područjima znanosti, uključujući robotiku, računalno umrežavanje, sigurnost, inženjerski dizajn, rudarstvo podataka, financije, ekonomiju i mnoga druga. U ovom radu je predstavljeno šest najnovijih prirodom inspiriranih metaheurističkih algoritama u računalnoj znanosti. Prvi algoritam pokušava što bolje pozicionirati lovca u prostoru za lov na jelene. Drugi algoritam određuje koliko ima mužjaka u jednoj skupini plavih majmuna. Treći algoritam određuje medvjeđi njuh kroz olfaktivne organe te se pokušava predvidjeti kvaliteta mirisa iz skupa komponenata mirisa. Četvrti algoritam predstavlja neobično ponašanje parenja škotskog jelena. Peti algoritam oponaša faze leta i napada (lova) surog orla. Šesti algoritam pokušava modelirati ponašanje *Agame* guštera u potrazi za hranom i njihov učinkovit način hvatanja plijena. Svi ti algoritmi su temeljeni na prirodnoj pojavi.

S druge strane baš zbog težnje ka jednostavnosti algoritama te pojednostavljivanja postavljanja problema, nemoguće je dostići kompleksnost s kojom priroda rješava probleme. U svakom slučaju, nema sumnje da je metaheuristika nadahnuta prirodom zaslužila svoje mjesto kao moćni alat za optimizaciju pa se stoga očekuje da će se ova grana znanosti nastaviti razvijati u bliskoj budućnosti.

Postoji mnogo stvari koje se mogu poboljšati u načinu na koji funkcionira metaheuristička zajednica. Prijelaz iz zajednice usmjerene na izvedbu, u zajednicu u kojoj je znanstveno razumijevanje važnije, dogodit će se u narednom razdoblju. Bez sumnje, to će dovesti do razvoja još boljih i još učinkovitijih metaheuristika, ali će također dovesti do heuristike koja se može koristiti izvan okruženja razvojnog laboratorija.

LITERATURA

[1] K. Sörensen, M. Sevaux i F. Glover, »A History of Metaheuristics Handbook of Heuristics,« 2018.

[2] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L. i Stein, Clifford, Introduction To Algorithms, London, England: The MIT Press, 2009.

[3] T. Chu, Human Purpose and Transhuman Potential, USA: Origin Press, 2014.

[4] L.J. Fogel, A.J. Owens i M.J. Walsh, Artificial Intelligence Through Simulated Evo-lution., New York: John Wiley & Sons, 1966..

[5] J. Holland, Adaptation In Natural and Artificial Systems, USA: University of Michigan Press, 1975.

[6] S. Kirkpatrick, C.D. Gelatt i M.P. Vecchi, Optimization by simulated annealing, SCIENCE, 1983.

[7] F. Glover, Tabu search—part i, ORSA Journal on computing, 1989.

[8] J. Hopfield, »Neural networks and physical systems with emergent collective computa-tional capabilities,« svez. 79, 1982.

[9] P. Oliveto, J. He i X. Yao, Time complexity of evolutionary algorithms for combi-natorial optimization: A decade of results, International Journal of Automation andComputing, 2007.

[10] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts, Caltech Concurrent Computation Program.

[11] J-P. Watson, L. Barbulescu, L.D. Whitley i A.E. Howe, Contrasting structuredand random permutation flow-shop scheduling problems: search-space topology andalgorithm performance, INFORMS Journal on Computing, 2002.

[12] M. Abdel-Basset, L. Abdel-Fatah i A. K. Sangaiah, »Metaheuristic Algorithms: A Comprehensive Review,« 2018.

[13] A.Galántai, »Journal of Computational and Applied Mathematics,« svez. 124, br. 1–2, December 2000.

[14] J. M. M. K. Sasan Harifi, »Giza Pyramids Construction: an ancient-inspired metaheuristic algorithm for optimization,« svez. 14, br. 4, 2021.

[15] M. A. N. Alfonso Ramos-Michel, Solving Reality-Based Trajectory Optimization Problems with Metaheuristic Algorithms Inspired by Metaphors, Springer, Cham, 2022.

[16] I. Rechenberg, Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution, Stuttgart: FrommanHolzboog.

[17] J. H. Holland, Adaptation in natural and artificial systems, The MIT Press, 1976.

[18] J. R. Koza, GENETIC PROGRAMMING: A PARADIGM FOR GENETICALLY, Stanford: Computer Science Department, 1990.

[19] R. Eberhart i J. Kennedy, A new optimizer using particle swarm theory, Nagoya, Japan: IEEE, 1995.

[20] M. Dorigo i M. Birattari, »Ant colony optimization,« svez. 1, br. 4, 2006.

[21] L. N. d. Castro i J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer Science & Business Media, 2002.

[22] N. Nanas i A. D. Roeck, Multimodal Dynamic Optimization: From Evolutionary Algorithms to Artificial Immune Systems, Springer, Berlin, Heidelberg: ICARIS, 2007.

[23] Z. Ji i D. Dasgupta, Revisiting Negative Selection Algorithms, MIT Press, 2007.

[24] L. N. d. Castro i F. J. V. Zuben, The Clonal Selection Algorithm with Engineering Applications, Proceedings of GECCO, 2000.

[25] H. J. C. B. Heder S. Bernardino, »Artificial Immune Systems for Optimization,« u *Nature-Inspired Algorithms for Optimisation*, Berlin, Heidelberg, Springer, 2009, pp. 389-411.

[26] R. K. J. &. B. Poli, Particle swarm optimization., 2007.

[27] Y.-J. Zheng, Water wave optimization: A new nature-inspired metaheuristic, China: Computers & Operations Research, 2015.

[28] M. Burnet, The clonal selection theory of acquired immunity, 1960.

[29] Albert Y. S. Lam i Victor O. K. Li, »Chemical-Reaction-Inspired Metaheuristic for Optimization,« svez. 14, br. 3, 2010.

[30] MarjanAbdechiri, Mohammad RezaMeybodi i Helena Bahrami, »Gases Brownian Motion Optimization: an Algorithm for Optimization (GBMO),« svez. 13, br. 5, 2013.

[31] Z. W. Geem i J. H. Kim, »A New Heuristic Optimization Algorithm: Harmony Search,« svez. 76, br. 2, 2001.

[32] Mora-Gutiérrez R.A., Ramírez-Rodríguez J. i Rincón-García E.A, »An optimization algorithm inspired by musical composition,« 2012.

[33] S. A. Salem, BOA: A novel optimization algorithm, Cairo, Egypt: IEEE, 2012.

[34] SeyedaliMirjalili, »SCA: A Sine Cosine Algorithm for solving optimization problems,« svez. 96, 2016.

[35] A. W. R. M. N. R. A. H. T. Nicholas Metropolis, »Equation of state calculations by fast computing machines,« svez. 21, br. 6, 1953.

[36] V. D. R.V.Rao, »Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems,« svez. 43, br. 3, 2011.

[37] A. H. Kashan, »League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships,« svez. 16, 2014.

[38] C. M. Fred Glover, »The general employee scheduling problem. An integration of MS and AI,« svez. 13, br. 5, 1986.

[39] P. N.Mladenović, »Variable neighborhood search,« svez. 24, br. 11, 1997.

[40] S. V. Éric D. Taillard, »Popmusic — Partial Optimization Metaheuristic under Special Intensification Conditions,« svez. 15, 2002.

[41] S. S. S. Binitha S, »A Survey of Bio inspired Optimization,« svez. 2, br. 2, 2012.

[42] F. F. A. G. Erik Cuevas, An Introduction to Nature-Inspired Metaheuristics and Swarm Methods, Springer, Cham, 2019.

[43] M. Mitchell, Genetic algorithms: an overview, Santa Fe , 1995.

[44] H. F. S. H.-P. Bäck Thomas, A Survey of Evolution Strategies., 1991.

[45] K. P. Rainer Storn, Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, J. Glob. Optim., 1997.

[46] B. L. Sette S., Genetic programming: principles and applications., 2001.

[47] X.-S. Yang, Nature-inspired Metaheuristic Algorithms, UK: Luniver Press, 2010.

[48] M. A. D. C. D. A. O. N. Erik Cuevas, Advances of Evolutionary Computation: Methods and Operators, Springer Cham, 2016.

[49] K. G. M. J. G. DIGALAKIS, »ON BENCHMARKING FUNCTIONS FOR GENETIC ALGORITHMS,« svez. 00, 2000.

[50] K. A. De Jong, AN ANALYSIS OF THE BEHAVIOR OF A CLASS OF GENETIC ADAPTIVE SYSTEMS., University of Michigan ProQuest Dissertations Publishing, 1975.

[51] L. D., Users Guide to the PGAPack Parallel Genetic Algorithm Library, Argonne National Laboratory, 1995.

[52] ,. S. P. N. N. P. R. R. B. R. D. B. G. BRAMMYA, Deer Hunting Optimization Algorithm:A New Nature-Inspired Meta-heuristic Paradigm, Oxford University Press, 2018.

[53] B. A.-K. Maha Mahmood, »The blue monkey: A new nature inspired metaheuristic optimization algorithm,« svez. 7, br. 3, 2019.

[54] G. B. A. B. Zewdu Kifle, »Population size, group composition and behavioural ecology of geladas (Theropithecus gelada) and human-gelada conflict in Wonchit Valley, Ethiopia,« svez. 16, br. 21, 2013.

[55] K. A. Ensermu Kibebew, »Population Status, Group Size, and Threat to Boutourlini's Blue Monkeys (Cercopithecus mitis boutourlinii) in Jibat Forest, Ethiopia,« svez. 07, br. 2, 2017.

[56] A. Ghasemi-Marzbali, »A novel nature-inspired meta-heuristic algorithm for optimization: bear smell search algorithm,« 2020.

[57] X.-C. H. Xiao-Ping Zhou, »Reliability analysis of slopes using UD-based response surface methods combined with LASSO,« svez. 233, 2018.

[58] J. J. H. Zhaoping Li, »Modeling the olfactory bulb and its neural oscillatory processings,« svez. 61, 1989.

[59] A. K. M. R. M. K. P. I. Kimberly J. Grossman, »Glomerular activation patterns and the perception of odor mixtures,« svez. 27, br. 10, 2008.

[60] Z. Li, »A model of the olfactory bulb and beyond,« 1989.

[61] M. H.-K. R. T.-M. Amir Mohammad Fathollahi-Fard, »Red deer algorithm (RDA): a new nature-inspired meta-heuristic,« svez. 24, 2020.

[62] S. R. T.H.Clutton-Brock, »The logical stag: Adaptive aspects of fighting in red deer (Cervus elaphus L.),« svez. 27, br. 1, 1979.

[63] »Red deer,« Wikimedia Foundation, Inc., 22 08 2022. [Mrežno]. Available: https://en.wikipedia.org/wiki/Red_deer. [Pokušaj pristupa 27 08 2022].

[64] K. E.McComb, »Female choice for high roaring rates in red deer, Cervus elaphus,« svez. 41, br. 1, 1991.

[65] F. C.R.Thouless, »Conflict between red deer hinds: the winner always wins,« svez. 34, br. 4, 1985.

[66] D. E. .. G. Brad L. Miller, »Genetic Algorithms, Tournament Selection,,« 1995.

[67] D. L. Adam Lipowski, »Roulette-wheel selection via stochastic acceptance,« svez. 391, br. 6, 2012.

[68] X.-S. Yang, »Firefly Algorithm, Stochastic Test Functions and Design Optimisation,« svez. 2, br. 2, 2010.

[69] M. D. N. A. A. Abdolkarim Mohammadi-Balani, »Golden Eagle Optimizer: A nature-inspired metaheuristic algorithm,« svez. 152, 2020.

[70] Anonymous, »Golden eagle,« Wikimedia Foundation, Inc., 5 7 2022. [Mrežno]. Available: https://en.wikipedia.org/wiki/Golden_eagle. [Pokušaj pristupa 5 8 2022].

[71] B. R. N. Z. H. B. B. C. F. J. D. Tack, »Ecosystem processes, land cover, climate, and human settlement shape dynamic distributions for golden eagle across the western US,« svez. 23, br. 1, 2020.

[72] N. S. D. P. V. Neetesh Kumar, » D.P. Artificial lizard search optimization (ALSO): a novel nature-inspired meta-heuristic algorithm.,« svez. 25, 2021.

[73] T. Y. M. E. C.-S. D. L. D. J. C. A. J. &. R. J. F. Thomas Libby, »Tail-assisted pitch control in lizards, robots and dinosaurs.,« svez. 481, 2012.

[74] G. BRAMMYA, S. PRAVEENA, N.S. NINU PREETHA, R. RAMYA, B.R. RAJAKUMAR i D. BINU, »Deer Hunting Optimization Algorithm: A New Nature-Inspired Meta-heuristic Paradigm,« 2019/05/24.

[75] A. Kashan, »League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships,« svez. 16, 2014.

[76] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, UK: Luniver Press, 2010.

[77] K. A. Ensermu Kibebew, Population Status, Group Size, and Threat to Boutourlini’s Blue Monkeys (Cercopithecus mitis boutourlinii) in Jibat Forest, Ethiopia, 2017.

POPIS SLIKA