SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Algoritam šišmiša i algoritam kukavičjeg pretraživanja

Velimir Kovačić
Voditelj: *Marin Golub*

Zagreb, svibanj 2024.

SADRŽAJ

1. Uvod

Metaheuristika se odnosi na oblikovni obrazac ugrađivanja prethodno stečenog znanja u izgradnju optimizacijskih algoritama. Smatra se potpodručjem stohastičke optimizacije. Stohastička optimizacija je kombinacija algoritama i slučajnih procesa radi pronalaženja optimalnih rješenja.

Prirodom nadahnute metaheuristike su one koje crpe ideje za pretraživanje prostora stanja iz prirode. Algoritmi rojeva su potkategorija prirodom nadahnutih metaheuristika. Svoja svojstva crpe iz kolektivnog ponašanja životinja i kukaca. Roj se sastoji od više individua čije je ponašanje vrlo jednostavno, ali zajedno mogu djelovati inteligentno. Postoje mnogi takvi algoritmi, najpoznatiji su mravlji algoritam i algoritam pčela.

U ovom seminaru razmatraju se manje poznati algoritmi: algoritam šišmiša i algoritam kukavičjeg pretraživanja. Oba su algoritma opisana i prikazan je njihov pseudokod. Navedeni su načini na koje se algoritmi mogu lokalno pokrenuti i dani su primjeri raznih primjena algoritama.

2. Algoritam šišmiša

2.1. Uvod

Algoritam šišmiša[18] (BA) metaheuristički je optimizacijski algoritam iz skupine algoritama rojeva čestica. Nadahnut je eholokacijom šišmiša.

Šišmiši su sisavci iz roda *Chiroptera* i ujedno jedini sisavci s krilima. Tvore oko 20% svih vrsta sisavaca. Dijele su 2 podreda: veliki šišmiši (*Macrochiroptera*) i mali šišmiši (*Microchiroptera*). Iako sve vrste koriste eholokaciju u nekoj mjeri, šišmiši iz reda malih šišmiša je koriste opsežnije.[19] Eholokacija kod malih šišmiša oblik je sonara, odašilju vrlo glasan i kratkotrajan zvučni impuls te primaju jeku koju je izazvao odbijajući se od okoline. Služi im za pronalaženje plijena, svojih legla i izbjegavanje prepreka u mraku.

Za većinu vrsta impuls traje između 5 i 20 ms i ima konstantnu frekvenciju u intervalu od 25 kHz do 150 kHz. U normalnim uvjetima taj impuls odašilju 10 do 20 puta u sekundi, dok ga prilikom lova mogu odašiljati i do 200 puta u sekundi. Ispostavlja se da šišmiši imaju izrazitu sposobnost obrade signala. Uz pretpostavku brzine zvuka od 340 m/s, može se odrediti da je valna duljina impulsa između 2mm i 14mm za frekvencijski raspon od 25 kHz do 150 kHz. Intenzitet zvuka može biti do 110 dB, a smanjuje se što je šišmiš bliže plijenu.

2.2. Algoritam

U samom algoritmu potrebno je idealizirati svojstva eholokacije:

1. Svi šišmiši koriste eholokaciju radi određivanja udaljenosti.
2. Znaju razliku između plijena i prepreka.
3. i-ti šišmiš leti brzinom $\vec{v_i}$, na položaju $\vec{x_i}$, s frekvencijom f_i i intenzitetom zvuka A_i.

4. Šišmiši namještaju frekvenciju zvučnih impulsa i učestalost odašiljanja $r \in [0, 1]$ prema udaljenosti od plijena.

5. Intenzitet zvuka je u intervalu $[A_{\text{min}}, A_0]$.

6. Frekvencija je u intervalu $[f_{\text{min}}, f_{\text{max}}]$.

Pretpostavlja se da postoji N šišmiša u prostoru dimenzije n. U svakoj iteraciji algoritma, svaki šišmiš izvodi 3 radnje:

1. Kretanje

2. Lokalna pretraga

3. Ažuriranje intenziteta zvuka i učestalosti odašiljanja

2.2.1. Kretanje

Kretanje je slično kretanju čestica u algoritmu roja čestica. U svakom koraku vektor brzine se usmjeri prema najboljem rješenju i odvije se kretanje šišmiša. Promjena vektora brzine ovisi o frekvenciji.

Položaj i-tog od N šišmiša u koraku t predstavlja n-dimenzionalni vektor:

$$\vec{x_i}^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t) \tag{2.1}$$

Brzinu i-tog od N šišmiša u koraku t predstavlja n-dimenzionalni vektor:

$$\vec{v_i}^t = (v_{i1}^t, v_{i2}^t, \ldots, v_{in}^t) \tag{2.2}$$

U koraku t računa se nova frekvencija:

$$f_i = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}})\beta \tag{2.3}$$

gdje je β realan broj uzorkovan iz uniformne razdiobe na intervalu $[0, 1]$.
U koraku t računa se nova brzina:

$$\vec{v_i}^t = \vec{v_i}^{t-1} + (\vec{x_i}^* - \vec{x_i}^{t-1})f_i \tag{2.4}$$

gdje je $\vec{x_i}^*$ položaj šišmiša s najvećom vrijednošću ciljne funkcije u koraku $t - 1$.
U koraku t računa se novi položaj:

$$\vec{x_i}^t = \vec{x_i}^{t-1} + \vec{v_i}^t \tag{2.5}$$

2.2.2. Lokalna pretraga

Nakon što su svi šišmiši završili s kretanjem, pokreće se lokalna pretraga za svakog šišmiša. Lokalna je pretraga nasumična. Položaj šišmiša se ažurira na sljedeći način:

$$\vec{x}_{\text{novi}} = \vec{x}_{\text{stari}} + \varepsilon A^t \tag{2.6}$$

gdje je ε slučajan broj iz intervala $[-1, 1]$, a A^t je srednja vrijednost intenziteta zvuka svih šišmiša.

2.2.3. Intenzitet zvuka i učestalost odašiljanja

Ako je iznos funkcije cilja u novom položaju šišmiša bolji od onog u prethodnom, novi se položaj prihvaća te se ažuriraju intenzitet zvuka i učestalost odašiljanja. U koraku t računa se novi intenzitet zvuka:

$$A_i^t = \alpha A_i^{t-1} \tag{2.7}$$

U koraku t računa se nova učestalost odašiljanja:

$$r_i^t = r_i^0[1 - e^{-\gamma(t-1)}] \tag{2.8}$$

Konstante γ i α su parametri algoritma. Oni određuju brzinu konvergencije. Za svaki $0 < \alpha < 1$ i $\gamma > 0$, vrijedi:

$$\lim_{t\to\infty} A_i^t = 0 \tag{2.9}$$

$$\lim_{t\to\infty} r_i^t = r_i^0 \tag{2.10}$$

Za $r_i = 1$ i $A_i = 0$, algoritam degradira u obični algoritam roja čestica.

2.2.4. Pseudokod

Algorithm 1 Algoritam šišmiša

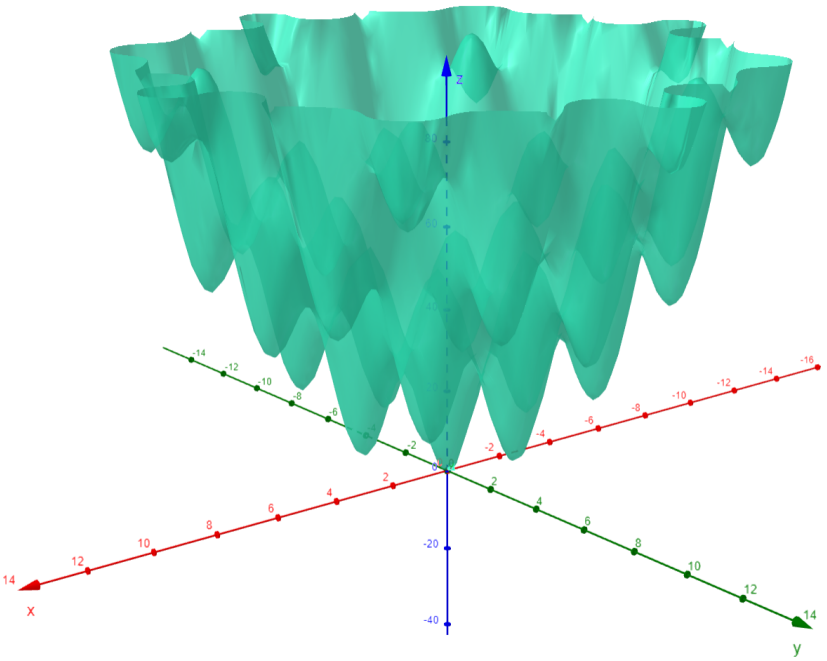
```
1:  function BA(obj(x), n, N, fmin, fmax, Amin, A0, maxIter, α, γ)
2:      x = INITX(n, d)
3:      v = INITV(n, d)
4:      f = INITF(fmin, fmax)
5:      A = INITA(Amin, A0)
6:      r0 = r = INITR(0,1)                          ▷ Inicijalizacija struktura podataka
7:      for iter = 1 to maxIter do
8:          for i = 1 to N do
9:              b = UNIFORMDIST(0, 1)
10:             f[i] = fmin + (fmax - fmin)*b
11:             v'[i] = v[i] + (x* - x[i]) * f[i]
12:             x'[i] = x[i] + v[i]                  ▷ Kretanje šišmiša
13:             rand = UNIFORMDIST(0, 1)
14:             if rand > r[i] then
15:                 x'[i] = x* + NORMALDISTVEC(n, 0, 1)
16:             e = UNIFORMDISTVEC(n, -1, 1)
17:             Amean = MEAN(A)
18:             x'[i] = x'[i] + Amean*e              ▷ Lokalna pretraga
19:             rand = UNIFORMDIST(0, 1)
20:             if rand < A[i] AND obj(x'[i]) < obj(x[i]) then
21:                 x[i] = x'[i]                     ▷ Prihvat novog rješenja
22:                 v[i] = v'[i]
23:                 A[i] = α * A[i]
24:                 r[i] = r0[i] * (1 - e**(-γ * iter))
25:         x* = MAXARG(obj(x))
26:     return x*
```

2.3. Programsko ostvarenje

2.3.1. Primjer funkcije za minimizaciju

Neka se minimizira n-dimenzionalna funkcija:

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2 + 25 \sum_{i=1}^{n} \sin^2 x_i \tag{2.11}$$



Slika 2.1: Graf funkcije $f(\vec{x})$ za $n = 2$

Funkcija ima minimum u točki $\vec{0}$ i on iznosi $f(\vec{0}) = 0$.

2.3.2. Programsko ostvarenje MEALPY

Algoritam šišmiša i mnogi drugi metaheuristički algoritmi programski su ostvareni u sklopu knjižnice programa otvorenog koda MEALPY[17] programskog jezika Python. Knjižnica se može preuzeti naredbom konzole:

```
> pip install mealpy
```

Kod 1: Naredba konzole za preuzimanje knjižnice MEALPY

Prethodno opisani problem funkcije (2.11) s $n = 10$ definira se u programu kao rječnik i pokreće se algoritam šišmiša s 1000 iteracija, brojem šišmiša $N = 50$, $\alpha = \gamma = 0.9$, $[A_{\text{min}}, A_0] = [1, 2]$, $[f_{\text{min}}, f_{\text{max}}] = [-10, 10]$. U razredu `BA` postoji više različitih inačica algoritma, najsličniji opisanom je `AdaptiveBA`, stoga je on korišten u primjeru.

```
import numpy as np
from mealpy import FloatVar, BA
from numpy import pi as PI

n = 10

def objective_function(solution):
    return np.sum(solution**2) + 25*np.sum(np.sin(solution)**2)

problem = {
    "obj_func": objective_function,
    "bounds": FloatVar(lb=(-2*PI,)*n, ub=(2*PI,)*n),
    "minmax": "min",
}


model = BA.AdaptiveBA(epoch=1000, pop_size=50,
                      loudness_min = 1.0, loudness_max = 2.0,
                      pf_min = -10., pf_max = 10.)

g_best = model.solve(problem)

print("Solution:", g_best.solution)
print("Fitness:", g_best.target.fitness)
```

Kod 2: Pokretanje optimizacije vlastite funkcije

```
Solution: [-0.07566663 -2.81959093 ...  0.12792114]
Fitness: 106.61031296676825
```

Ispis 1: Ispis primjera

2.4. Algoritam šišmiša u primjeni

2.4.1. Modeliranje dinamike bioloških sustava

Algoritam šišmiša korišten je za prilagodljiv odabir vrijednosti parametara modela za rekonstrukciju dinamike bioloških sustava.[11] Uspješnost metode je pokazana na primjeru procjene parametara dinamike endocitoze.

2.4.2. Procjena položaja ljudskog tijela

Praćenje položaja ljudskoj tijela iz snimaka u laboratorijskim uvjetima postavljeno je kao 31-dimenzionalni optimizacijski problem.[1] Pokazano je da algoritam šišmiša radi bolje od drugih srodnih algoritama (poput algoritma roja čestica.)

2.4.3. Grupiranje

Razvijen je algoritam grupiranja temeljen na algoritmu K-sredina i algoritmu šišmiša.[10] Algoritam ne zahtijeva definiranje hiperparametra K, nego sredine pronalazi korištenjem algoritma šišmiša. Prednost je ubrzanje brzine konvergencije algoritma šišmiša i smanjivanje ovisnosti algoritma K-sredina o početno odabranim sredinama.

2.4.4. Problem ekonomske raspodjele opterećenja

U problemu ekonomske raspodjele opterećenja, cilj je zadovoljiti potrebe za energijom uz minimalne financijske troškove. Optimizacija termalne elektrane provedena je algoritmom šišmiša uz rezultate bolje od rezultata drugih srodnih algoritama.[2]

2.4.5. Podudaranje slika

Problem podudaranja slika u računalnom vidu je poistovjećivanje slika istih objekata slikanih u različitim uvjetima.[20] Algoritam šišmiša, uz dodatak mutacija, ko-

rišten je za rješavanje ovog problema. Pokazano je da je takav algoritam na ovom problemu učinkovitiji od običnog algoritma šišmiša.

2.4.6. Odabir značajki

Binarna inačica algoritma šišmiša korištena je za odabir optimalnog podskupa značajki.[12] Pristup spaja istraživačku moć šišmiša i brzinu klasifikatora *Optimum-Path Forest* za maksimizaciju točnosti na skupu za provjeru. Pokusima je utvrđeno da takav pristup ima bolje performanse od drugih poznatih algoritama.

2.4.7. Optimizacija sustava rezervoara

Algoritam šišmiša upotrebljen je za optimizaciju rezervoarskog sustava Karoun-4 u Iranu.[3] Algoritam je bio učinkovitiji od linearnog programiranja, nelinearnog programiranja i genetskog algoritma.

2.4.8. Online učenje unaprijednih neuronskih mreža

Online učenje slojeva unaprijednih neuronskih mreža provedeno je algoritmima: BA, GA i PSO.[9] Usporednom analizom bilo je vidljivo da algoritam šišmiša ima bolje performanse od ostalih.

3. Algoritam kukavičjeg pretraživanja

3.1. Uvod

Algoritam kukavičjeg pretraživanja[7] (CSA) metaheuristički je optimizacijski algoritam utemeljen na parazitskom ponašanju nekih vrsta kukavica i slučajnoj šetnji (Lévyjevom letu.)

3.1.1. Kukavičji parazitizam

Kukavice su ptice iz porodice *Cuculidae*, jedine porodice u redu *Cuculiformes*. Mnoge vrste kukavica nesu jaja u gnijezda drugih ptica. To ponašanje se naziva *parazitiranje legla*. Neke vrste kukavica evoluirale su tako da im jaja veličinom i bojama sliče jajima vrste domaćina. Time je vjerojatnost da će ptica domaćin izbaciti njihova jaja manja, a reproduktivna moć samih kukavica veća. Uz to, kukavice se liježu ranije i instinktivno izbacuju ostala jaja iz gnijezda, povećavajući si vjerojatnost opstanka.

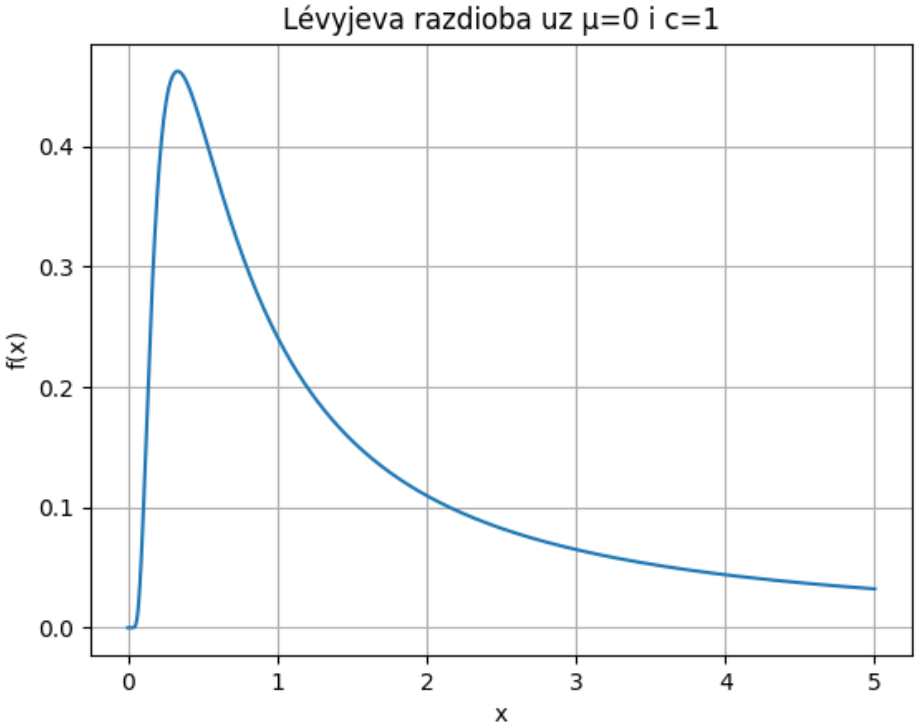
3.1.2. Lévyjev let

U prirodi, životinje traže hranu na slučajan ili kvazi-slučajan način. Lévyjev let je slučajna šetnja u kojoj je duljina koraka iz Lévyjeve razdiobe ili neke druge stabilne razdiobe. Razdioba je stabilna ako je linearna kombinacija dvije nezavisne jednako distribuirane slučajne varijable, koje prate tu razdiobu, također prati tu razdiobu do na parametre. Funkcija gustoće Lévyjeve razdiobe je:

$$f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{\frac{3}{2}}} \tag{3.1}$$

gdje je μ parametar lokacije, a c parametar skaliranja. Za $c = 1$ i $\mu = 0$ (bez skaliranja i pomaka), funkcija gustoće je:

$$f(x) = \frac{1}{\sqrt{2\pi}}\frac{e^{-\frac{1}{2x}}}{x^{\frac{3}{2}}} \tag{3.2}$$

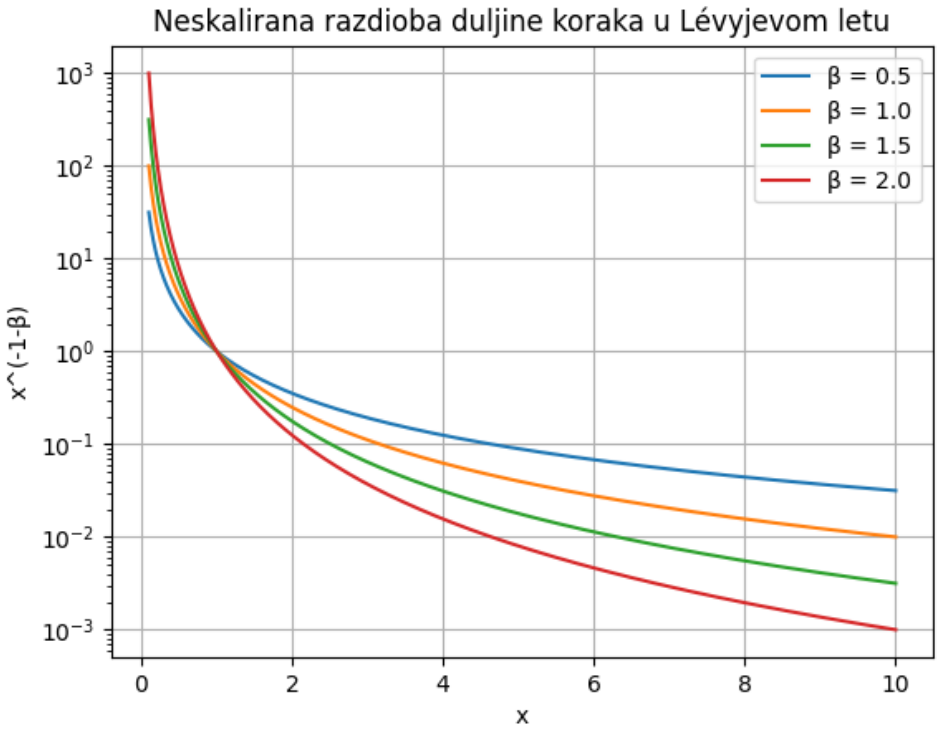


Slika 3.1: Graf Lévyjeve razdiobe za $\mu = 0$ i $c = 1$

Radi jednostavnosti, za određivanje duljine koraka s u Lévyjevom letu, koristi se sljedeća stabilna razdioba:

$$L(s) \sim |s|^{-1-\beta} \tag{3.3}$$

gdje je parametar $\beta \in \langle 0, 2]$. Prije upotrebe tu je razdiobu potrebno skalirati.



Slika 3.2: Graf neskalirane razdiobe za Lévyjev let na logaritamskoj skali

3.2. Algoritam

U samom algoritmu potrebno je idealizirati ponašanje kukavica:

1. Kukavice liježu po jedno jaje u nasumično odabrano gnijezdo.
2. Gnijezda s dobrim jajima (rješenjima) prenose se na sljedeću generaciju.
3. Broj domaćina je fiksan.
4. Domaćin može otkriti strano jaje s vjerojatnošću P_a.
5. Ako domaćin otkrije strano jaje, izbacuje ga ili gradi novo gnijezdo na drugoj lokaciji.

Pretpostavlja se N kukavičjih gnijezda u prostoru dimenzije n. U svakoj iteraciji algoritma, slučajno odabrana kukavica izvodi 2 radnje:

1. Izlegne se i nasumično leti (izvodi Lévyjev let)
2. Zamjenjuje slučajno odabrano gnijezdo, ako je na njenom položaju vrijednost funkcije cilja veća

Nakon što se N puta odabere slučajna kukavica, udio od Pa najlošijih gnijezda se izbacuje. Točnije, modificiraju se Lévyjevim letovima.

3.2.1. Kretanje

Položaj i-tu od N kukavica u koraku t predstavlja n-dimenzionalni vektor:

$$\vec{x_i}^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t) \tag{3.4}$$

Lévyjev let za i-tu od N kukavica u koraku t odvija se na sljedeći način:

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \alpha \oplus L(\beta) \tag{3.5}$$

gdje je α faktor skaliranja, a operator $\oplus$ predstavlja međusobno množenje pripadnih elemenata vektora.

3.2.2. Pseudokod

Algorithm 2 Algoritam kukavičjeg pretraživanja

```
1:  function CSA(obj(x), n, N, Pa, α, β, maxIter)
2:      x = INITX(n, d)                          ▷ Inicijalizacija struktura podataka
3:      for iter = 1 to maxIter do
4:          for k = 1 to N do
5:              i = RANDINT(0, N)                ▷ Slučajan odabir kukavice
6:              x[i] = x[i] + α ⊕ L(β)            ▷ Lévyjev let
7:              j = RANDINT(0, N)                ▷ Slučajan odabir gnijezda
8:              if obj(x[i]) > obj(x[j]) then
9:                  x[j] = x[i]
10:         x = SORTBY(x, obj)                   ▷ Uzlazno sortiranje po funkciji cilja
11:         for i = 1 to Pa*N do
12:             x[i] = x[i] + α ⊕ L(β)           ▷ Najlošija gnijezda se "izbacuju"
13:         x* = MAXARG(obj(x))
14:     return x*
```

3.3. Programsko ostvarenje

3.3.1. Programsko ostvarenje MEALPY

Algoritam kukavičjeg pretraživanja ostvaren je u sklopu knjižnice programa otvorenog koda MEALPY[17] programskog jezika Python. Knjižnica je detaljnije opisana u pododjeljku 2.3.2.

Prethodno opisani problem funkcije (2.11) s $n = 10$ definira se u programu kao rječnik i pokreće se algoritam kukavičjeg pretraživanja s 1000 iteracija, brojem šišmiša $N = 50$ i vjerojatnošću otkrivanja $P_a = 0.3$.

```
import numpy as np
from mealpy import FloatVar, CSA
from numpy import pi as PI

n = 10

def objective_function(solution):
    return np.sum(solution**2) + 25*np.sum(np.sin(solution)**2)

problem = {
    "obj_func": objective_function,
    "bounds": FloatVar(lb=(-2*PI,)*n, ub=(2*PI,)*n),
    "minmax": "min",
}

model = CSA.OriginalCSA(epoch=1000, pop_size=50, p_a = 0.3)

g_best = model.solve(problem)

print("Solution:", g_best.solution)
print("Fitness:", g_best.target.fitness)
```

Kod 3: Pokretanje optimizacije vlastite funkcije

```
Solution: [-0.12270494 -0.92564155 ...  2.82223478]
Fitness: 73.53109251662303
```

Ispis 2: Ispis primjera

3.4. Algoritam kukavičjeg pretraživanja u primjeni

3.4.1. Optimizacija strukture automobilskih dijelova

Algoritam kukavičjeg pretraživanja primijenjen je na problem problem optimizacije strukturnog dizajna automobilskih dijelova.[5] Pokazano je da algoritam pronalazi dobra rješenja.

3.4.2. Pronalazak rubova slika

Algoritam je upotrebljen za optimizaciju parametara antecedensa za sustav za pronalaženje rubova utemeljen na Sobelovoj tehnici i neizrazitim intervalima.[8] Problem je složen zbog uporabe neizrazitih intervala, stoga se koriste heurističke metode. Zaključeno je da je algoritam kukavičjeg pretraživanja učinkovit za ovakve probleme.

3.4.3. Pronalazak štete na mostovima

Pronalazak štete na mostovima i gredama problem je koji se rješavao umjetnim neuronskim mrežama.[16] Učenje težina neuronske mreže algoritmom kukavičjeg pretraživanja bilo je brže i točnije od učenja evolucijskim algoritmom.

3.4.4. Modeliranje toka goriva zrakoplova

Cilj jednog rada bio je stvaranje modela jačine toka goriva prilikom uspona zrakoplova korištenjem algoritma kukavičjeg pretraživanja.[14] Jačina toka funkcija je visine i brzine, a korišteni su podaci stvarnog zrakoplova. Takav model bio je u stanju predvidjeti jačinu toka s visokom točnošću.

3.4.5. Diskretni optimizacijski problemi

Diskretna inačica algoritma kukavičjeg pretraživanja primijenjena je na niz poznatih diskretnih optimizacijskih problema: TSP, JSSP i QAP.[15] Dobiveni su rezultati usporedivi s onima srodnih algoritama.

3.4.6. Fuzija slika

Fuzija slika je postupak prikupljanja i spajanja bitnih podataka iz više slika. Razvijen je algoritam utemeljen na algoritmu *Grey Wolf* i algoritmu kukavičjeg pretraživanja s rezultatima boljim od ostalih metaheurističkih algoritama.[6]

4. Zaključak

Oba algoritma (algoritam šišmiša i algoritam kukavičjeg pretraživanja), unatoč vrlo jednostavnim formulacijama, pokazuju obećavajuće rezultate na raznim problemima od znanstvenih i matematičkih do industrijskih i komercijalnih.

Prednost algoritma šišmiša je mogućnost ugađanja omjera između istraživačkog i eksploatacijskog ponašanja šišmiša putem parametara eholokacije. Nadalje, jednostavno ga je ostvariti i nije računalno zahtjevan. Algoritam je već poznat te se može pronaći u različitim knjižnicama optimizacijskih algoritama, poput MEALPY, što olakšava primjenu. Pokazuje se korisnim u primjeni na različite optimizacijske probleme.

Algoritam kukavičjeg pretraživanja, kao i algoritam šišmiša, nije računalno zahtjevan i jednostavno ga je ostvariti. Može se pronaći u brojnim knjižnicama optimizacijskih algoritama, uključujući i MEALPY. Može se primijeniti na optimizacijske probleme u različitim domenama s dobrim rezultatima.

Općenito, prirodnom nadahnuti metaheuristički algoritmi rojeva na učinkovit način pronalaze neko rješenje problema. Takvo rješenje neće nužno biti najbolje moguće, ali može biti dovoljno dobro za pojedino područje primjene. Daljnjim istraživanjem mogu se pronaći učinkovitiji algoritmi i poboljšati postojeći.

5. Literatura

[1] Shakeel Akhtar, Abdul-Rahim Ahmad, Eihab Abdel-Rahman, i Tariq Naqvi. A metaheuristic bat-inspired algorithm for full body human pose estimation. *Proceedings of the 2012 9th Conference on Computer and Robot Vision, CRV 2012*, 05 2012. doi: 10.1109/CRV.2012.55.

[2] Sandeep Biswal, Ajit Barisal, Aurobindo Behera, i Tapan Prakash. Optimal power dispatch using bat algorithm. stranice 1018–1023, 04 2013. ISBN 978-1-4673-6149-1. doi: 10.1109/ICEETS.2013.6533526.

[3] Omid Bozorg-Haddad, Iman Karimirad, Samaneh Seifollahi-Aghmiuni, i Hugo A. Loáiciga. Development and application of the bat algorithm for optimizing the operation of reservoir systems. *Journal of Water Resources Planning and Management*, 141(8):04014097, 2015. doi: 10.1061/(ASCE)WR.1943-5452.0000498. URL https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29WR.1943-5452.0000498.

[4] Mridul Chawla i Manoj Duhan. Bat algorithm: A survey of the state-of-the-art. *Applied Artificial Intelligence*, 29(6):617–634, 2015. doi: 10.1080/08839514.2015.1038434. URL https://doi.org/10.1080/08839514.2015.1038434.

[5] Ismail Durgun i Ali Yildiz. Structural design optimization of vehicle components using cuckoo search algorithm. *Materials Testing*, 54:185–188, 03 2012. doi: 10.3139/120.110317.

[6] Sayantan Dutta i Ayan Banerjee. Optimal image fusion algorithm using modified grey wolf optimization amalgamed with cuckoo search, levy fly and mantegna algorithm. stranice 284–290, 03 2020. doi: 10.1109/ICIMIA48430.2020.9074959.

[7] Amir Gandomi, Xin-She Yang, i Amir Alavi. Cuckoo search algorithm: a meta-

heuristic approach to solve structural optimization problems. *Engineering With Computers*, 29:245–245, 04 2013. doi: 10.1007/s00366-012-0308-4.

[8] C. I. Gonzalez, J. R. Castro, P. Melin, i O. Castillo. Cuckoo search algorithm for the optimization of type-2 fuzzy image edge detection systems. U *2015 IEEE Congress on Evolutionary Computation (CEC)*, stranice 449–455, 2015. doi: 10.1109/CEC.2015.7256924.

[9] Koffka Khan i Sahai Ashok. A comparison of ba, ga, pso, bp and lm for training feed forward neural networks in e-learning context. *International Journal of Intelligent Systems and Applications*, 4, 06 2012. doi: 10.5815/ijisa.2012.07.03.

[10] Dr Komarasamy G i Amitabh Wahi. An optimized k-means clustering technique using bat algorithm. *European Journal of Scientific Research*, 84:263–273, 08 2012.

[11] Jiann-Horng Lin, Chao-Wei Chou, Chorng-Horng Yang, Hsien-Leing Tsai, i I-Ho Lee. A bio-inspired optimization algorithm for modeling the dynamics of biological systems. stranice 206–211, 09 2012. ISBN 978-1-4673-2838-8. doi: 10.1109/IBICA.2012.58.

[12] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa, i X.-S. Yang. Bba: A binary bat algorithm for feature selection. U *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, stranice 291–297, 2012. doi: 10.1109/SIBGRAPI.2012.47.

[13] Anand Nayyar, Dac-Nhuong Le, i Nhu Gia Nguyen. *Advances in swarm intelligence for optimizing problems in computer science*. CRC press, 2018.

[14] Rıdvan Oruç i Tolga Baklacioglu. Modelling of fuel flow-rate of commercial aircraft for the climbing flight using cuckoo search algorithm. 02 2020. doi: 10.1108/AEAT-10-2019-0202.

[15] Aziz Ouaarab. *Discrete Cuckoo Search for Combinatorial Optimization*. Springer Singapore, 2020. ISBN 9789811538360. doi: 10.1007/978-981-15-3836-0. URL `http://dx.doi.org/10.1007/978-981-15-3836-0`.

[16] H. Tran-Ngoc, S. Khatir, G. De Roeck, T. Bui-Tien, i M. Abdel Wahab. An efficient artificial neural network for damage detection in bridges and beam-like structures by improving training parameters using

cuckoo search algorithm. *Engineering Structures*, 199:109637, 2019. ISSN 0141-0296. doi: https://doi.org/10.1016/j.engstruct.2019.109637. URL https://www.sciencedirect.com/science/article/pii/S0141029619308351.

[17] Nguyen Van Thieu i Seyedali Mirjalili. Mealpy: An open-source library for latest meta-heuristic algorithms in python. *Journal of Systems Architecture*, 2023. doi: 10.1016/j.sysarc.2023.102871.

[18] Xin-She Yang. *A New Metaheuristic Bat-Inspired Algorithm*, stranice 65–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-12538-6. doi: 10.1007/978-3-642-12538-6_6. URL https://doi.org/10.1007/978-3-642-12538-6_6.

[19] Leksikografski zavod Miroslav Krleža. Netopiri. *Hrvatska enciklopedija*, 2013. – 2024. URL https://www.enciklopedija.hr/clanak/netopiri.

[20] Jia Zhang i Gai-Ge Wang. Image matching using a bat algorithm with mutation. *Applied Mechanics and Materials*, 203:88–93, 10 2012. doi: 10.4028/www.scientific.net/AMM.203.88.

6. Sažetak

Algoritam šišmiša i algoritam kukavica metaheuristički su optimizacijski algoritmi nadahnuti procesima iz prirode. U ovom radu dan je pregled samih algoritama, načini na koje se mogu izvesti na računalu i njihove primjene.