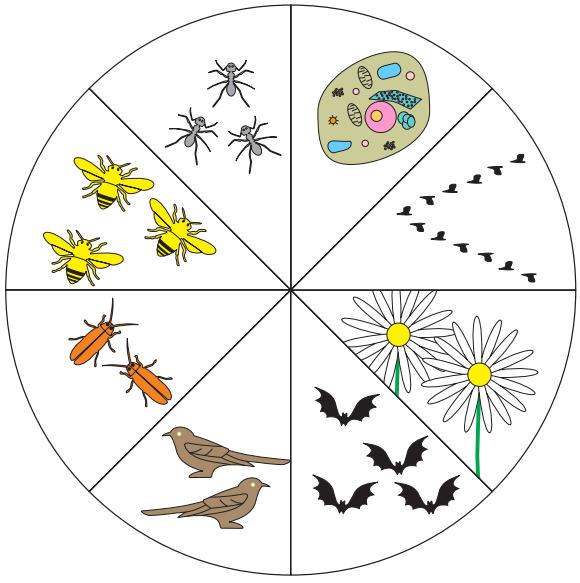
# Algoritem na osnovi obnašanja netopirjev

Karin Ljubič in Iztok Fister ml.

→ **Algoritmi po vzorih iz narave so posebna vrsta računalniških algoritmov za reševanje težkih optimizacijskih problemov v računalništvu, matematiki, medicini in tudi v industriji [3]. Do danes so znanstveniki razvili že veliko tovrstnih algoritmov, med njimi najbolj izstopajo vzori mravelj, čebel, kresnic in tudi kukavic.**



**SLIKA 1.**
Algoritmi po vzorih iz narave

Eden najnovejših predstavnikov teh algoritmov je algoritem, ki deluje na osnovi obnašanja netopirjev [4, 5]. Netopirji spadajo med sesalce in so edini sesalci z letalno mreno; ta jim omogoča letenje (ne samo jadranje). Med letenjem večina netopirjev »cvilka«. To je posebno zvočno valovanje, ki ga zaradi visokih frekvenc imenujemo tudi ultrazvočno valovanje. Ti ultrazvočni valovi se odbijajo od predmetov v okolici in tako potujejo nazaj k netopirjem. Netopirji tako slišijo odmeve svojih poslanih glasov in na podlagi gostote prejetih odmevov določijo, kako blizu je ovira. Če je ovira zelo blizu, se bo oddani zvočni signal vrnil zelo hitro in z gosto frekvenco. Če pa je ovira dlje, se bo oddani zvočni signal vrnil kasneje in z redkejšo frekvenco, kakor je bil oddan. Ta mehanizem zaznavanja ovir med letom s pomočjo ultrazvočnih valov se imenuje eholokacija. Poleg netopirjev se s pomočjo eholokacije orientirajo tudi nekatere druge živalske vrste.

- Delfini: Ultrazvočne valove uporabljajo za navigacijo, komunikacijo, lov in obrambo v temnih vodah.
- Kiti: Ultrazvočne valove proizvajajo s premikanjem zraka med zračnimi prostori (sinusi) v glavi. Uporabljajo ga za detektiranje hrane in podvodno navigacijo.
- Rovke: Rovke so majhni sesalci. Ultrazvočne valove uporabljajo pri gibanju v svojem naravnem habitatu, ne toliko za plenilstvo in iskanje hrane.

Postavlja se vprašanje, kako omenjeni mehanizem eholokacije uporabiti v računalništvu. Odgovor na to vprašanje je našel profesor Xin-She Yang leta 2010, ko je tedaj deloval na univerzi v Cambridgu. Ustvaril je zelo zanimiv, preprost in učinkovit algoritem. Algoritem je povezal z vzori iz narave ter eholokacije in zasnoval tri idealizirana pravila:

1. Vsi netopirji uporabljajo eholokacijo za spremljanje razdalje do tarčnih objektov.
2. Netopirji letijo naključno s hitrostjo $v_i$ na pozicijo $x_i$ s fiksno frekvenco $Q_{min}$, variabilno valovno dolžino $\lambda_i$ in glasnostjo $A_i$ v iskanju plena. Samodejno lahko prilagajajo valovno dolžino (ali frekvenco) svojih oddanih pulzov in prilagajo raven oddajanja pulzov $r_i \in [0, 1]$ glede na bližino tarče.

3. Glasnost variira od največje (pozitivne) $A_0$ do minimalne $A_{min}$ vrednosti konstante.

Psevdokod algoritma je predstavljen v spodnjem zapisu (1) in sestoji iz naslednjih glavnih korakov:

- Vhod algoritma predstavlja populacija netopirjev, ki je predstavljena z realnimi števili. Izhod pa predstavlja najboljša rešitev ter njena pripadajoča vrednost.
- Inicializacija začetne populacije netopirjev se izvede s pomočjo naključnega generatorja realnih števil.
- Generiranje nove rešitve se izvede s premikom navideznih netopirjev glede na naslednje enačbe [4]:

$$\begin{aligned} Q_i^{(t)} &= Q_{min} + (Q_{max} - Q_{min})N(0,1), \\ \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^t + (\mathbf{x}_i^t - \mathbf{best})Q_i^{(t)}, \\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \end{aligned} \tag{1}$$

  kjer je $N(0,1)$ naključno število.
- Lokalno iskanje skrbi za izboljšanje najboljše rešitve z uporabo hevristike RWDE (random walk with direct exploitation – naključni sprehod z neposrednim izkoriščanjem).
- Ocenjevanje najboljše rešitve in shranjevanje najboljše rešitve glede na verjetnost (parameter A določa verjetnost shranjevanja najboljše rešitve).
- Iskanje najboljše rešitve.

Delovanje si lahko še ogledamo na praktičnem primeru, kjer moramo najti minimum funkcije Sphere: $F(\vec{x}_i) = \sum_{i=1}^{n} x_i^2$ in pri kateri iščemo rešitev v prostoru $-100{,}00 \leq x_i \leq 100{,}00$.

Ta funkcija ima minimum pri 0

- $(\mathbf{f}(\mathbf{x_1}, \ldots, \mathbf{x_n}) = \mathbf{f}(\mathbf{0}, \ldots, \mathbf{0}) = \mathbf{0})$.

Na začetku določimo parametre algoritma, kot so velikost populacije $NP$, parameter $A$ (glasnost), parameter $r$ (raven pulza), parameter $Qmin$ (frekvenčni minimum), parameter $Qmax$ (frekvenčni maksimum) ter dimenzijo problema $D$. Obširen opis parametrov je opisan v [4]. Potem uvedemo naključno populacijo z naključnim generatorjem, ki generira vrednost med podanima mejama. V našem primeru bo to med $-100$ in 100. Zatem ocenimo populacijo ter poiščemo najboljšo rešitev. Ocenitev poteka tako, da vstavimo vrednosti v eni populaciji v funkcijo, ki jo optimiramo. V našem primeru bo boljša tista, ki bo bližja minimumu funkcije. Zatem sledi glavna zanka, ki teče do ustavitvenega pogoja (npr. število generacij). Znotraj te zanke teče še ena zanka, ki gre skozi vse populacije. Tukaj se generirajo rešitve s pomočjo prej predstavljenih formul. Novo rešitev torej dobimo tako, da vzamemo rešitev iz prejsnje generacije in seštejemo s hitrostjo, ki smo jo izračunali v tej generaciji. Po generiranju nove rešitev sledi korak lokalnega iskanja, kjer poskušamo s pomočjo hevristike RWDE še malo izboljšati rešitev. To poteka s pomočjo enačbe

- $$\mathbf{y^{(t)}} = \mathbf{x}^* + \epsilon A_i^{(t)} \mathbf{v}_i^{(t)}, \tag{2}$$

kjer $N(0,1)$ označuje naključno generirano število z intervala $[-1,1]$, $\epsilon$ je skalirni faktor, $\mathbf{x}^*$ je trenutno najboljša rešitev in $A_i^{(t)}$ je glasnost.

Zatem korakom sledi ocenjevanje nove rešitve ter pogojno shranjevanje najboljše rešitve. Na koncu pa poiščemo najboljšo rešitev.

**Algoritem 1** Algoritem na osnovi obnašanja netopirjev

**Vhod:** Populacija netopirjev $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \ldots Np$, $MAX\_FE$.
**Izhod:** Najboljša rešitev $\mathbf{x}_{best}$ in njena pripadajoča vrednost $f_{min} = \min(f(\mathbf{x}))$.

```
1: inicializiraj();
2: eval = oceni_novo_populacijo();
3: f_min = najdi_najboljšo_rešitev(x_best);
4: while ustavitveni_pogoj_ni_dosežen do
5:   for i = 1 to Np do
6:     y = generiraj_novo_rešitev(x_i);
7:     if rand(0,1) > r_i then
8:       y = izboljšaj_najboljšo_rešitev(x_best)
9:     end if { korak lokalnega iskanja }
10:    f_new = oceni_novo_rešitev(y);
11:    eval = eval + 1;
12:    if f_new ≤ f_i and N(0,1) < A_i then
13:      x_i = y; f_i = f_new;
14:    end if { pogojno shrani najboljšo rešitev }
15:    f_min = najdi_najboljšo_rešitev(x_best);
16:  end for
17: end while
```

Če na kratko povzamemo, vidimo, da ima ta algoritem nekatere značilnosti sposojene od ostalih algoritmov po vzorih iz narave. Zelo je podoben algoritmu rojev delcev [1] in hibridiziran s hevristiko RWDE ter simuliranim ohlajanjem [2]. Prva skrbi v tem algoritmu za izboljševanje najboljše rešitve, medtem ko druga skrbi za raznolikost populacije. Lahko bi tudi rekli, da je lokalno iskanje povezano s komponento izkoriščanja, medtem ko je simulirano ohlajanje povezano s komponento preiskovanja v procesu iskanja tega algoritma netopirjev. Izkoriščanje se v algoritmu kontrolira s parametrom **r** in preiskovanje s parametrom **A**. Algoritem na osnovi obnašanja netopirjev je bil do sedaj uporabljen v veliko aplikacijah v realnem svetu. Iz velikega nabora literature lahko vidimo, da so raziskovalci uporabili ta algoritem pri problemih obdelave slik, filtriranju neželene pošte, različnih vrstah razporejanja, podatkovnega rudarjenja in še veliko več. V prihodnje lahko pričakujemo še večjo uporabo tega algoritma ter nove modifikacije algoritma, ki bi se še bolj uspešno spopadale z različnimi problemi.

## Literatura


[1] J. Kennedy in R. Eberhart, *Particle swarm optimization, In Neural Networks*, Proceedings, IEEE International Conference on, volume 4, 1942–1948, IEEE, 1995.

[2] S. Kirkpatrick, D. G. Jr. in M. P. Vecchi, *Optimization by simmulated annealing*, Science 220(4598), 671–680, 1983.

[3] X.-S. Yang, *Nature-inspired metaheuristic algorithms*, Luniver Press, 2010.

[4] X.-S. Yang, *A new metaheuristic bat-inspired algorithm*, In Nature inspired cooperative strategies for optimization (NICSO 2010), 65–74, Springer, 2010.

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




# Barvni sudoku

→ V 8 × 8 kvadratkov moraš vpisati začetna naravna števila od 1 do 8 tako, da bo v vaki vrstici, v vsakem stolpcu in v kvadratkih iste barve (pravokotnikih 2 × 4) nastopalo vseh 8 števil.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 7 | | | 6 | | 1 | |
| | | 1 | | | | 5 | 2 |
| 8 | | | | | | | |
| | 2 | | | 5 | 7 | | 3 |
| | | 2 | | | | | |
| | 6 | 3 | | | | | |
| | | | 1 | | | 6 | |
| | 3 | 8 | | | 1 | 2 | |

## REŠITEV BARVNI SUDOKU

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | 2 | 1 | 4 | 7 | 8 | 3 | 6 |
| 7 | 6 | 8 | 3 | 1 | 5 | 4 | 2 |
| 1 | 7 | 5 | 2 | 8 | 3 | 6 | 4 |
| 4 | 3 | 6 | 8 | 5 | 2 | 1 | 7 |
| 3 | 8 | 7 | 5 | 4 | 6 | 2 | 1 |
| 6 | 4 | 2 | 1 | 3 | 7 | 5 | 8 |
| 2 | 5 | 4 | 7 | 6 | 1 | 8 | 3 |
| 8 | 1 | 3 | 6 | 2 | 4 | 7 | 5 |