TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Bat algorithm

12.11.2022

Zadanie č.2/7

- Algoritmus netopiera je metaheuristický globálny optimalizačný algoritmus . Inšpiroval sa echolokačným správaním netopierov s rôznou pulzovou frekvenciou a objemom. Algoritmus Bat vyvinul Xin-She Yang v roku 2010.

- Globálna optimalizácia je odvetvie aplikovanej matematiky a numerickej analýzy, ktoré sa snaží nájsť globálne minimá alebo maximá funkcie alebo množiny funkcií na danej množine.

- Metaheuristické algoritmy, ako napríklad optimalizácia pomocou roja častíc, algoritmus Firefly a hľadanie harmónie, sa v súčasnosti stávajú výkonnými metódami na riešenie mnohých náročných optimalizačných problémov. V tejto práci navrhujeme novú metaheuristickú metódu, algoritmus netopiera, ktorý je založený na echolokačnom správaní netopierov.

- Pri simuláciách musíme používať virtuálne netopiere. Musíme definovať pravidlá, ako sa aktualizujú ich polohy a a rýchlosti e v d-rozmernom priestore hľadania.

Pseudokód

Initializing the bat population X_i and V_i,with (i=1,2,...,m)

Objective function f(x),x=$(x^1, x^2, \dots, x^n)$

Define pulse frequency f_i

Initialize pulse rates r_i and the loudness A_i

While (t < Max number of iterations)

    For every bat b_i of the group, do

        Create new solutions by adjusting frequency, updating velocities and positions

        [using Eq. (1-3)]

        if (rand > r_i),do

            Choose one of the best solutions randomly and then create a

            local solution around it

        end if

        Create a new solution through random flying

        if (rand < A_i & f ($\bar{X}^k$) < f (X_i^k))

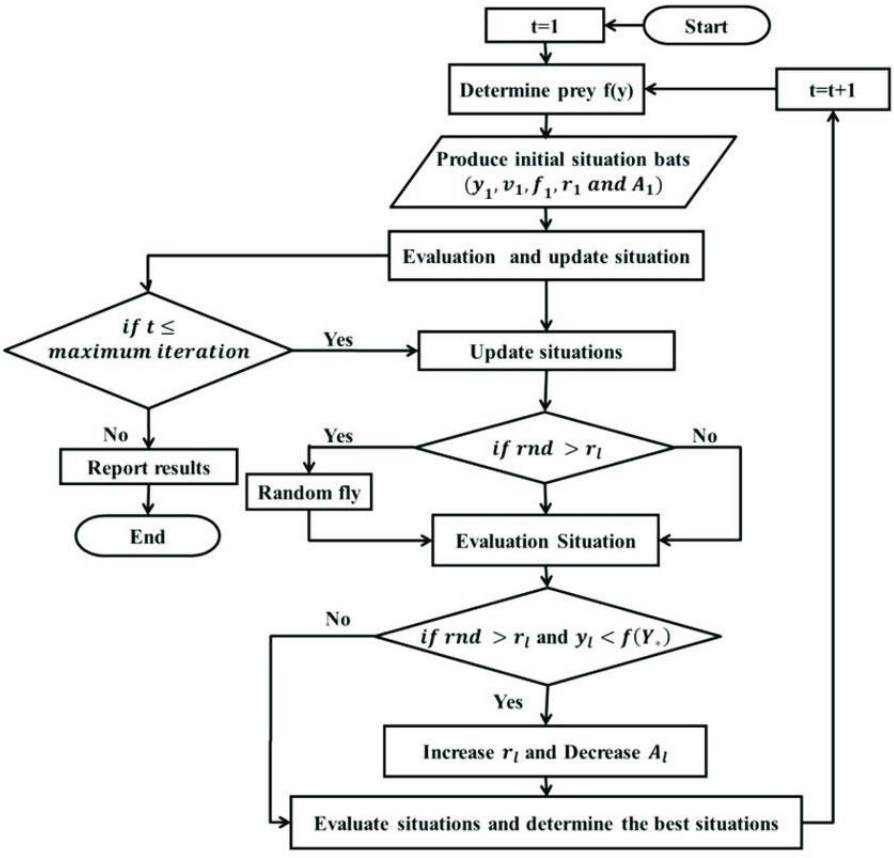
            Accept the solutions

            Increase r_i , reduce A_i

        end if

    Rank the bats then find the current best $\bar{X}$

end while



Problémy, ktoré možno riešiť pomocou tohto algoritmu.

- Kombinatorická optimalizácia a plánovanie
- Inverzné problémy a odhad parametrov
- Klasifikácie, zhlukovanie a dolovanie dát
- Spracovanie obrazu
- Fuzzy logika a iné aplikácie

1. Z hľadiska výpočtovej zložitosti možno spojité optimalizačné problémy považovať za jednoduché, hoci ich riešenie môže byť stále veľmi náročné.Kombinatorické problémy však môžu byť naozaj ťažké, často ťažké v nedeterministickom polynomiálnom čase Ramesh et al. prezentovali podrobnú štúdiu kombinovaných problémov ekonomického zaťaženia a emisného dispečingu pomocou netopierieho algoritmu.
2. Yang a i. používajú netopierí algoritmus naštudovanie optimalizácie topologického tvaru v mikroelektronickýchaplikáciách tak, aby sa materiály s rôznymi tepelnýmivlastnosťami mohli umiestniť tak, aby bol prenos tepla čo najefektívnejší pri prísnych obmedzeniach.

   Možno ho použiť aj na vykonanie odhadu parametrov ako inverzného problému.
   Ak sa dá inverzný problém správne sformulovať, potom môže netopierí algoritmus poskytnúť lepšie výsledky

3. Komarasamy a Wahi skúmali zhlukovanie K-meanklustering pomocou bat algoritmu a dospeli k záveru, že kombináciou K-means a BA možno dosiahnuť vyššiu efektívnosť, a tak dosahuje lepšie výsledky ako iné algoritmy.Khan a kol. prezentovali štúdiu zhlukovacieho problému pre kancelárske pracoviská pomocou fuzzy bat algoritmu. Khan a Sahari (2012a) tiež prezentovali štúdiu porovnania netopierieho algoritmu s PSO, GA a inými algoritmami v kontexte e-learningu, a tak naznačili, že netopierí algoritmus má jednoznačne určité výhody oproti iným algoritmom.

   Damodaram a Valarmathi (2012) skúmali detekciu phishingových webových stránok pomocou modifikovaného bat algoritmu a dosiahli veľmi dobré výsledky.

4. Du a Liu prezentovali variant netopierieho algoritmu s mutáciou na porovnávanie obrazov a naznačili, že ich model založený na netopierovi je efektívnejší a uskutočniteľnejší pri porovnávaní obrazov ako iné modely, napríklad diferenciálna evolúcia a genetické algoritmy.

5. Reddy a Manoj predstavili štúdiu optimálneho umiestnenia kondenzátorov na zníženie strát v distribučných systémoch pomocou netopierieho algoritmu. V kombinácii s fuzzylogic našli optimálne veľkosti kondenzátorov tak, aby sa minimalizovali straty.

Popis ukážkového problému

Netopiere vysielajú hlasné ultrazvukové vlny a počúvajú ozvenu, ktorá sa odráža od okolitých predmetov. Algoritmus netopiera používa pre jednoduchosť niekoľko zbožštených pravidiel.

(1) Netopiere používajú echolokáciu na vnímanie koristi, predátora alebo akýchkoľvek prekážok na ceste a vo vzdialenosti.
(2) Netopiere letia rýchlosťou a polohou . Majú frekvenciu f a hlasitosť, aby dosiahli svoju korisť. Môžu upravovať frekvenciu vysielania impulzov r.
(3) Keď sa priblížia ku koristi, pulz sa zvyšuje a hlasitosť sa znižuje.

Pomôžeme našim netopierom nájsť potravu a použijeme na to tento algoritmus.
Povedzme, že máme kolóniu netopierov a pomocou algoritmu presúvame netopiere a hľadáme netopiera, ktorý je najbližšie k potrave.
Samotný algoritmus je opísaný vyššie.

Popis experimentov a výsledkov

Parametre nášho algoritmu(D, M, N, A, r, freqMin, freqMax, Lower, Upper, function)

- self.D = D; # dimensions
- self.M = M; # population size
- self.N = N; # number of itterations
- self.A = A; # loudness
- self.r = r; # pulse rate
- self.freq = [0] * self.M; # frequency
- self.v = [[0 for i in range(self.D)] for j in range(self.M)]; # velocity
- self.Sol = [[0 for i in range(self.D)] for j in range(self.M)]; # Current solution (coordinates)
-
-
- self.freqMin = freqMin; # frequency min
- self.freqMax = freqMax; # frequency max
- self.Lower = Lower; # lower bound
- self.Upper = Upper; # upper bound
-
- self.f_min = 0.0; # minimum fitness
-
- self.Fitness = [0] * self.M; # fitness
- self.best = [0] * self.D; # closest bat
- self.Fun = function; # sum of squared coordinates

Vstupné parametre nášho algoritmu:

```
def Fun(D, sol):
    val = 0.0;
    for i in range(D):
        val = val + sol[i] * sol[i];
    return val;
```

1) BatAlgorithm(2, 10, 1000, 0.5, 0.5, 0.0, 2.0, -10.0, 10.0, Fun);
2) BatAlgorithm(1, 30, 1000, 0.7, 0.5, 0.0, 2.0, -10.0, 10.0, Fun);

Výsledky: (1 iteracia -> jeden pohyb netopiera, ak sú súradnice optimálnejšie)

1. 2.2003083996920496e-15 -> Fitness
2. 5.0870693423994995 -> Fitness