Bat algorithm - algorytm nietoperzowy wśród stochastycznych metod optymalizacji

Jakub Więckowski

17 Marzec 2020

Spis treści

Wprowadzenie

Intro i

Algorytm nietoperzowy to jeden z algorytmów probabilistycznych optymalizacji globalnej. Swoją nazwę i zasadę działania zawdzięcza zjawisku echolokacji u nietoperzy. Każdy z nich emituje dźwięk z określoną częstotliwością i poziomem głośności.

Algorytm ten został przedstawiony przez **Xin-She Yang** (mężczyzna) w **2010** roku [4], więc jest to stosunkowo młody algorytm. Mimo tego znalazł swoich zwolenników i jest używany w wielu problemach optymalizacji, takich jak globalna optymalizacja inżynierska [1], optymalizacja wielozadaniowa [2], czy też przy ograniczonych zadaniach optymalizacyjnych [3].

Algorytm

Algorytm i

Fala dźwiękowa, emitowana przez nietoperze pozwala im na odnalezienie ofiary, czy swojego położenia w przestrzeni nawet w całkowitej ciemności. Analogicznie do tego postępuje algorytm, w którym nietoperz reprezentowany jest przez pojedynczy stan. Każdy z nich posiada wektor prędkości oraz wektor pozycji. Dodatkowo każdy nietoperz ma zdefiniowaną charakterystyczną dla siebie częstotliwość, długość i głośność emitowanej fali dźwiękowej.

W trakcie pracy algorytmu te parametry zmieniają się wraz z poruszaniem się stanu imitującego zachowanie nietoperza. Podczas zbliżania się do optimum wzrasta puls, a głośność emitowanego dźwięku spada.

Algorytm ii

Pseudokod algorytmu nietoperzowego prezentuje się następująco:

1. Initialize the bat population $x_i (i = 1, 2, ..., n)$
2. Define pulse frequency f_i at x_i
3. Initialize pulse rates r_i and the loudness A_i
4. **While** (i < maximum number of iterations)
5. Generate new solutions by adjusting frequency and updating velocities and locations
6. **If** (rand $> r_i$)
7. Select a solution among the best solutions
8. Generate a local solution around the selected best solution
9. **End If**

Algorytm iii

10. Generate new solutions by flying randomly
11. **If** (rand $< A_i$ & $f(x_i) < f(x_best)$)
12. Accept the new solutions
13. Increase r_i and reduce A_i
14. **End If**
15. Rank the bast and find the current best x_best
16. **End While**

Wskazania dla doboru współczynników

Dobór współczynników

Definiując wartości początkowe współczynników do pracy algorytmu, można sugerować sie tym, że przy ograniczonej ilości iteracji, algorytm uzyska lepsze wyniki przy większej ilości nietoperzy [6].

Natomiast najistotniejszym parametrem, na który wpływ ma osoba implementująca algorytm jest częstotliwość wydawanego dźwięku [5]. Większe wartości częstotliwości pozwalają na uzyskanie lepszy rezultatów. Preferowane są również niższe początkowe wartości dla wskaźnika tętna, natomiast większe wartości dla prędkości. Parametr głośności nie ma znaczącego wpływu na osiągane rezultaty [5, 6].

Przedstawienie procedur

Procedury i

Listing 1: Bat Algorithm in Matlab

```
% Main programs starts here
function [best,fmin,N_iter]=bat_algorithm(para)

% Default parameters
if nargin<1, para=[20 1000 0.5 0.5]; end
n=para(1);      % Population size, typically 10 to 40
N_gen=para(2);  % Number of generations
A=para(3);      % Loudness  (constant or decreasing)
r=para(4);      % Pulse rate (constant or decreasing)
% This frequency range determines the scalings
% You should change these values if necessary
```

Procedury ii

```
Qmin=0;          % Frequency minimum
Qmax=2;          % Frequency maximum

% Iteration parameters
N_iter=0;        % Total number of function evaluations
% Dimension of the search variables
d=10;            % Number of dimensions
% Lower limit/bounds/ a vector
Lb=-2*ones(1,d);
% Upper limit/bounds/ a vector
Ub=2*ones(1,d);

% Initializing arrays
Q=zeros(n,1);    % Frequency
```

Procedury iii

```
v=zeros(n,d);   % Velocities
% Initialize the population/solutions
for i=1:n,
  Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);
  Fitness(i)=Fun(Sol(i,:));
end

% Find the initial best solution
[fmin,I]=min(Fitness);
best=Sol(I,:);

% Start the iterations - Bat Algorithm %
for t=1:N_gen,
% Loop over all bats/solutions
```

Procedury iv

```
for i=1:n,
  Q(i)=Qmin+(Qmin-Qmax)*rand;
  v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);
  S(i,:)=Sol(i,:)+v(i,:);
  % Apply simple bounds/limits
  Sol(i,:)=simplebounds(Sol(i,:),Lb,Ub);
  % Pulse rate
  if rand>r
  % The factor 0.001 limits the step sizes of random w
    S(i,:)=best+0.001*randn(1,d);
  end
  % Evaluate new solutions
  Fnew=Fun(S(i,:));
  % Update if the solution improves, or not too loud
```

Procedury v

```
      if (Fnew<=Fitness(i)) & (rand<A) ,
        Sol(i,:)=S(i,:);
        Fitness(i)=Fnew;
      end
      % Update the current best solution
      if Fnew<=fmin,
        best=S(i,:);
        fmin=Fnew;
      end
    end %end for
    N_iter=N_iter+n;
  end %end for

% Output/display
```

Procedury vi

```
disp(['Number of evaluations: ',num2str(N_iter)]);
disp(['Best = ',num2str(best),' fmin = ',num2str(fmin)]);

% Application of simple limits/bounds
function s=simplebounds(s,Lb,Ub)
  % Apply the lower bound vector
  ns_tmp=s;
  I=ns_tmp<Lb;
  ns_tmp(I)=Lb(I);
  % Apply the upper bound vector
  J=ns_tmp>Ub;
  ns_tmp(J)=Ub(J);
  % Update this new move
  s=ns_tmp;
```

Procedury vii

```
function z=Fun(u)
%
% ł przykadowa funkcja celu
%
%%%% ============ end ==========
```

Kod zaczerpnięty ze strony mathworks.com [13].
Wizualizacja pracy algorytmu

Opis zmiennych

Opis zmiennych

Zmienne pojawiające się przy pracy z algorytmem [5], które można modyfikować przy ustalaniu stanu początkowego:

- częstotliwość *F* - frequency
- głośność *L* - loudness
- wskaźnik tętna *PR* - pulse rate
- prędkość *V* - velocity
- pozycja *P* - position
- wymiarowość funkcji *FD* - function dimension
- ilość nietoperzy *N*

Wykorzystanie

Wykorzystanie

Algorytm nietoperzowy znalazł zastosowanie w nastepujących dziedzinach:

- globalna optymalizacja inżynierska [1, 8],
- optymalizacja wielozadaniowa [2],
- ograniczone zadania optymalizacyjne [3],
- optymalizacja w przestrzeni ciągłej [7],
- planowanie treningu sportowego [9],
- rozpoznawanie cyfr pisanych ręcznie [10].

Porównanie z innymi heurystykami

Porównanie

Literatura podaje przykłady, w których to algorytm nietoperzowy porównywany jest z Firefly Algorithm, Cuckoo Search [11] oraz z Particle Swarm Optimization [12].

Bibliografia

Bibliografia i

- Yang, X. S., & Gandomi, A. H. (2012). Bat algorithm: a novel approach for global engineering optimization. Engineering computations.
- Yang, X. S. (2012). Bat algorithm for multi-objective optimisation. arXiv preprint arXiv:1203.6571.
- Gandomi, A. H., Yang, X. S., Alavi, A. H., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. Neural Computing and Applications, 22(6), 1239-1255.
- Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65-74). Springer, Berlin, Heidelberg.

Bibliografia ii

- Carvalho, I. A., da Rocha, D. G., Silva, J. G. R., da Fonseca Vieira, V., & Xavier, C. R. (2017, July). Study of parameter sensitivity on bat algorithm. In International Conference on Computational Science and Its Applications (pp. 494-508). Springer, Cham.
- Lucilo, J. A., Pilar-Arceo, C. P., & Mendoza, E. R. (2016). On the Application of Bat Algorithm to Parameter Estimation of S-system Models. In Proceedings of the 16th Philippine Computing Science Congress (pp. 114-122).
- Chakri, A., Khelif, R., Benouaret, M., & Yang, X. S. (2017). New directional bat algorithm for continuous optimization problems. Expert Systems with Applications, 69, 159-175.

Bibliografia iii

- Yılmaz, S., & Küçüksille, E. U. (2015). A new modification approach on bat algorithm for solving optimization problems. Applied Soft Computing, 28, 259-275.
- Fister, I., Rauter, S., Yang, X. S., Ljubič, K., & Fister Jr, I. (2015). Planning the sports training sessions with the bat algorithm. Neurocomputing, 149, 993-1002.
- Tuba, E., Tuba, M., & Simian, D. (2016). Handwritten digit recognition by support vector machine optimized by bat algorithm.
- Arora, S., & Singh, S. (2013, August). A conceptual comparison of firefly algorithm, bat algorithm and cuckoo search. In 2013 International Conference on Control, Computing, Communication and Materials (ICCCCM) (pp. 1-4). IEEE.

Bibliografia iv

- Jiang, H., Wang, J., Wu, J., & Geng, W. (2017). Comparison of numerical methods and metaheuristic optimization algorithms for estimating parameters for wind energy potential assessment in low wind regions. Renewable and Sustainable Energy Reviews, 69, 1199-1217.
- https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm-demo?focused=5236333tab=function

Koniec ☺