Eksploracja przestrzeni rozwiązań funkcji Ackleya wykorzystując algorytm nietoperza

NMO_2021/22 Kacper Gałek (116772), Maksim Shchukin (77934)

1 Cel pracy

Celem pracy jest wykazać skuteczność algorytmu nietoperza, wykorzystując go do szukania ekstremum globalnego funkcji Ackleya.

1.1 Funkcja Ackleya

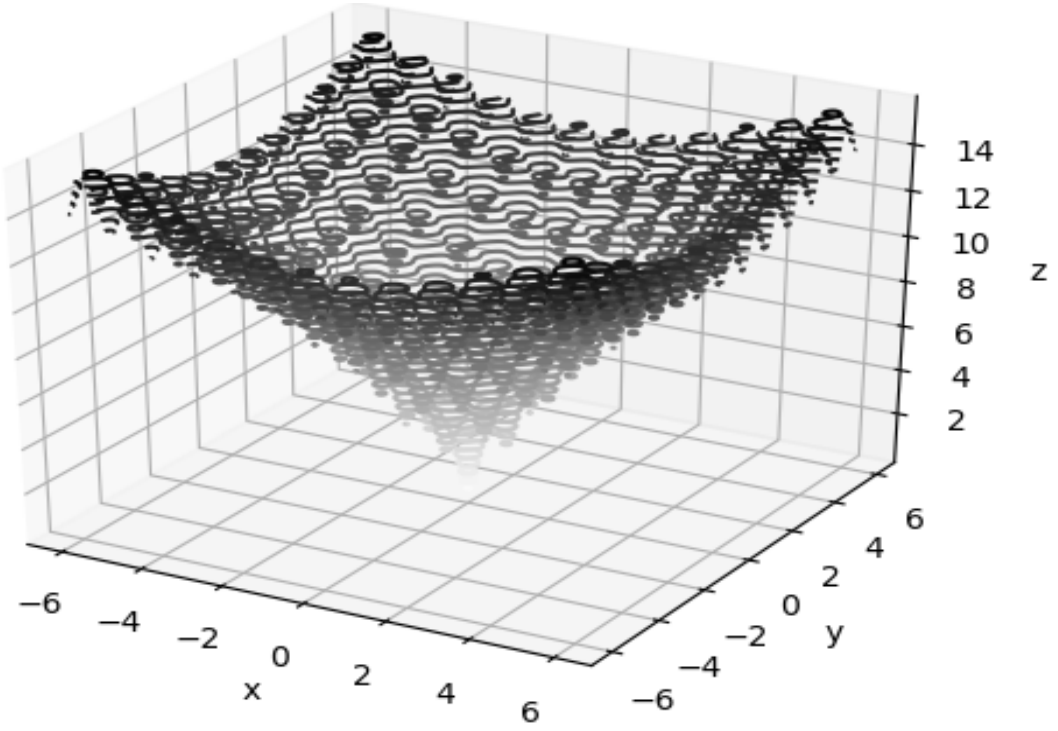
Funkcje zaproponował w 1987 doktor David Ackley. Wykorzystywana jest do testowania i weryfikacji działania algorytmów optymalizacyjnych.

1.2 Wzór i własności

Funkcja dana jest wzorem:

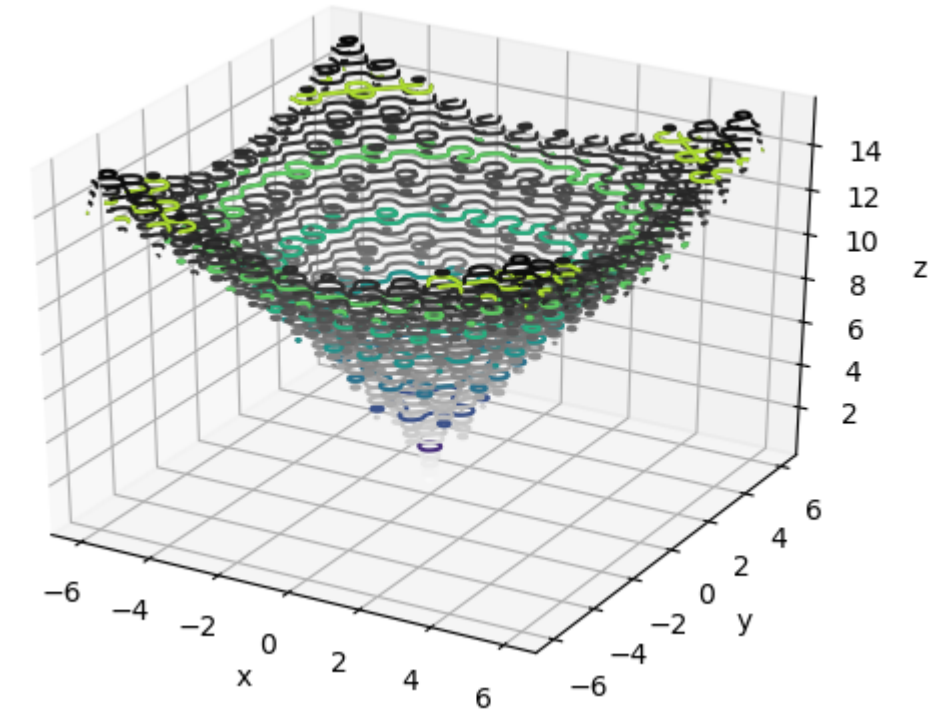
$$f(x,y) = -20 * e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5(cos(2\pi x)+cos(2\pi y))} + e + 20$$

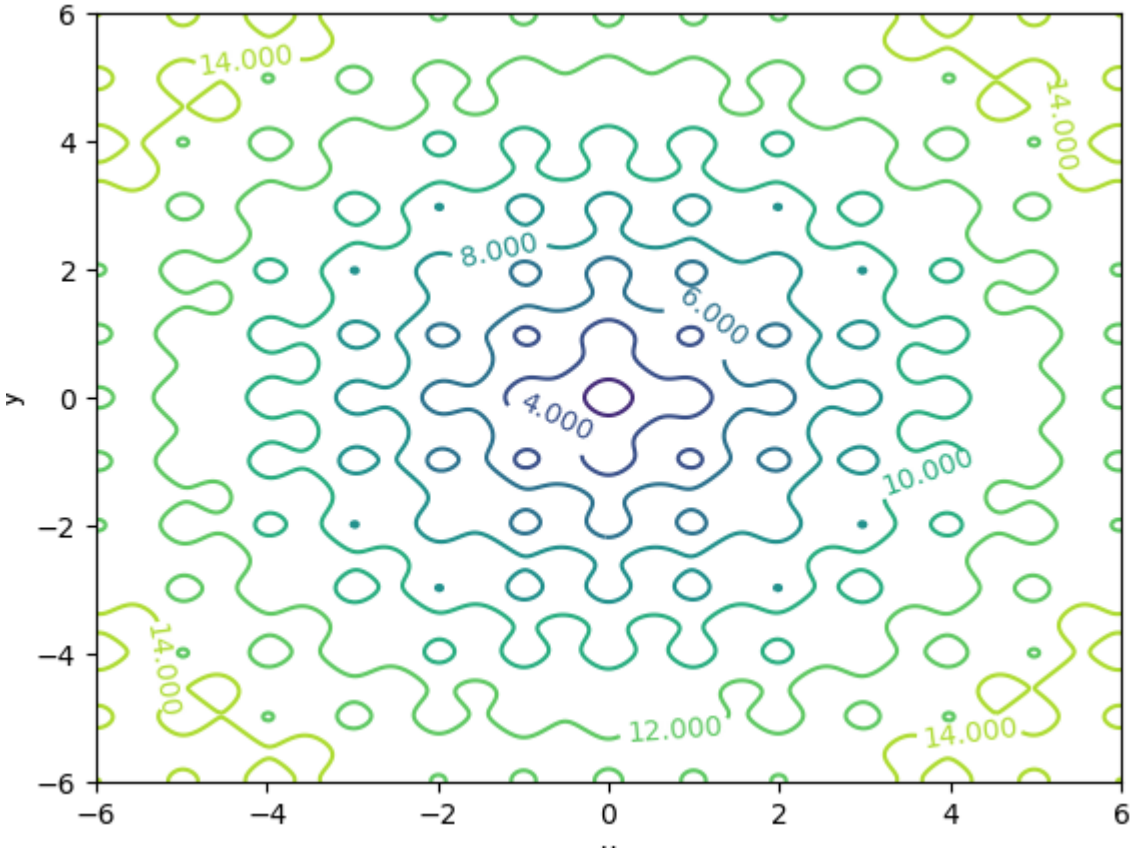
Jest to funkcja dwóch zmiennych osiągająca ekstremum globalne (minimum) dla **f(0,0) = 0**. Funkcja utrzymuje stały proporcjonalny kształt formujący lejek.



Rys 1. Wizualizacja funkcji Ackleya (źródło: Opracowanie własne)

Widoczne ‘wiry’ to ekstrema lokalne funkcji. Jest ich na tyle dużo, aby znacznie ograniczyć skuteczność metod lokalnych/zachłannych. Wizualizacja warstwicy funkcji dla otoczenia ekstremum globalnego (w przykładzie przyjęto obszary z zakresu <-6,6> obu zmiennych) prezentuje się następująco:





Tak wygenerowana mapa posłuży do śledzenia nietoperzy w okolicy optimum globalnego.

2 Algorytm Nietoperza (Bat Algorithm)

2.1 Przyrodnicza inspiracja dla algorytmu

Algorytm zaproponowany w 2010 roku przez doktora Xin-She Yang wzoruje się na zjawisku występującym w stadzie nietoperzy poszukujących żywności - wykorzystaniu echolokacji. Echolokacja pomaga nietoperzom w dokładnej orientacji w ciemności, znajdowaniu pożywienie przy jednoczesnym omijaniu przeszkód. Większość nietoperzy posługujących się zjawiskiem echolokacji wytwarza ultradźwięki w zakresie częstotliwości od 20 do 80 kHz, często bardzo ‘głośnych’ lecz niesłyszalnych dla człowieka. (Knypiński,2017)

2.2 Podstawy działania

Algorytm stara się poprzez wykorzystanie charakterystycznych dla nietoperzy cech połączyć cechy metod PSO i SA. Opiera się na koncepcji stada które przeszukuje przestrzeń rozwiązań, zna położenie najlepszego nietoperza, ale zmienia preferencje swoich zachowań w czasie (z iteracji na iterację). Ponadto na potrzeby algorytmu przyjęto bazowe założenia:

- Nietoperz potrafi rozpoznać przeszkodę (trzyma się przestrzeni rozwiązań)
- Nietoperz potrafi momentalnie zmienić częstotliwość wydawanych dźwięków
- Na początku polowania nietoperze są bardzo głośne $(\forall A_i^0 > 1)$

2.3 Przebieg algorytmu

Przebieg wariantu algorytmu nietoperza wykorzystany z pracy został zaproponowany przez Jianqiang Huang (Huang, 2020).

1. Inicjalizacja puli nietoperze
2. nadanie indywidualnych:
    a. lokalizacji $\vec{x}$
    b. prędkości v
    c. głośności A
    d. pulsowania r
    e. granicznego pulsu r0
    f. częstotliwości f
3. Wyznaczenie najlepszego nietoperza spośród losowo rzuconych na przestrzeń rozwiązań (na podstawie funkcji celu)

W pętli

4. aktualizacja częstotliwości prędkości i lokalizacji:
    f$inew$ = fmin + (fmax − fmin) · $Rand$(0, 1)
    $vi\ new$ = $vi\ old$ + ($x$old − x best) · fi. (x best to lokalizacja nietoperza z najlepszym dopasowaniem)
5. Określenie nowej pozycji (candidate solution)
    $xi\ new$ = $xiold$ + $vi\ new$
6. **jeżeli** rand(0,1) > r_i :

    Aktualny kandydat zostaje porzucony na korzyść kandydata z sąsiedztwa najlepszego rozwiązania w aktualnej iteracji według formuły
    $xnew$ = $xbest$ + $Rand$(−1, 1) · $\langle a \rangle$ gdzie $\langle a \rangle$ stanowi średnią głośność A rozwiązań w iteracji (A maleje wraz z iteracjami zawężając sąsiedztwo (krok 7)
7. **jeżeli** A_i > rand(0,1) & f($xi\ new$) < f(xi old):

    /Kandydat wchodzi do następnej iteracji, zachowuje f_i i v_i ,a A_i i r_i zmieniają się według formuł:
    A$inew$ = A$iold$ · α,
    ri=r0*(1 $- e^{-\gamma * k}$) gdzie k to liczba reprezentująca aktualną iterację
    Dla odpowiednio dobranych α i γ $\lim_{k \to \infty} r0 * (1 - e^{-\gamma * k}) = r0$ oraz A→ 0

    /W przeciwnym wypadku rozwiązanie przechodzi do następnej iteracji z nowymi wartościami f_i i v_i.
8. Ustalenie najlepszego rozwiązania dla kolejnej iteracji analogicznie do kroku 3.

2.4 Parametry wejściowe i oczekiwany rezultat

Aby zastosować algorytm nietoperza należy określić kilka kluczowych parametrów odnoszących się do cech indywidualnych nietoperzy oraz ustaleń dla całego stada. Zastosowane w symulacji parametry zostały opisane w poniższej tabeli:

Tabela 1 Parametry

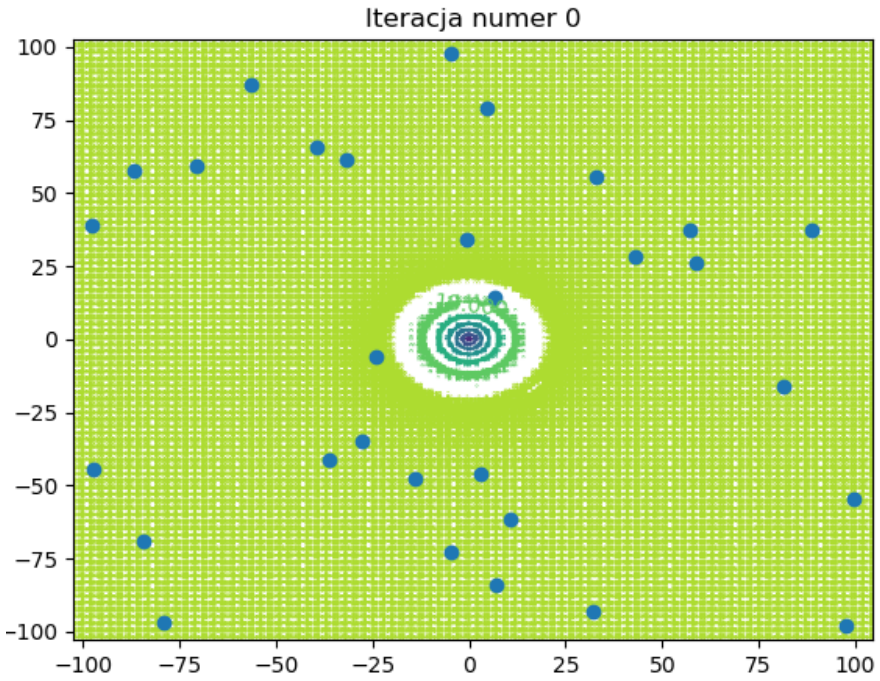
Parametry globalne		
Parametr	**Wartość**	**Własności/opis**
liczba iteracji	101	wpływa na długość trwania algorytmu oraz wyniki zastosowania kroku (7)
wielkość populacji	30	liczba nietoperzy “wrzucona” na przestrzeń rozwiązań
liczba wymiarów	2	jak długi będzie wektor pozycji nietoperza (ile zmiennych optymalizujemy)
zakres dla zmiennych	< -100,100 >	$x_k \in <-100, 100>$, $f: R^2$
częstotliwość minimalna fmin	0	Częstotliwość decyduje o tym jak duży udział w wektorze prędkości będzie miał wektor odległości od aktualnie najlepszego rozwiązania. Przyjęto zakres 0-1 z racji na stosunkowo niewielki rozmiar przestrzeni rozwiązań
częstotliwość maksymalna fmax	1	
stała alfa α	0.85	Reguluje tempo spadku A z kroku (7)
stała gamma γ	0.3	Reguluje tempo wzrostu r_i z kroku (7)
Parametry inicjalizacji (indywidualne dla nietoperza)		
pozycja początkowa x_i	[Rand(-100,100), Rand(-100,100)]	Wektor wylosowany na podstawie danych
wektor prędkości v_i	Rand(1,2)	Przedział <1,2> zaproponowany w oryginalnej publikacji metody
częstotliwość początkowa f_i	$Rand(0, 1)$	wzór z kroku (4) dla (0,1) $fi = fmin + (fmax - fmin) \cdot Rand(0, 1)$ fi=0+(1-0)*Rand(0,1)=Rand(0,1)
puls rate graniczny r_{0i}	Rand <0.75,1>	Wartość do której zbliża się r nietoperza w toku iteracji
puls rate początkowy r_i	r0*$(1 - e^{-gamma})$	Określane według wzoru (7)
głośność A_i	Rand(20,50)	Losowe wartości głośności nietoperza. Determinują jednocześnie restrykcyjność w przyjmowaniu rozwiązań i zakres lokalnego poszukiwania wokół najlepszego rozwiązania.

3 Wyniki

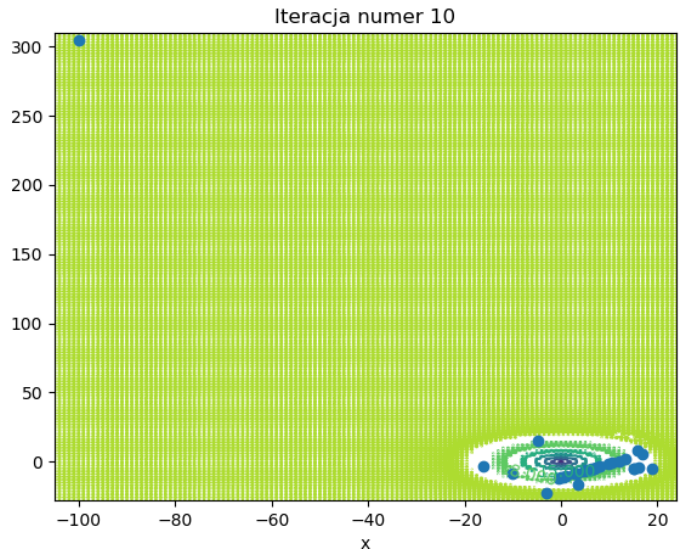
Po określeniu globalnych parametrów dla algorytmu stworzony został kod języka Python, realizujący kroki 1-8.

3.1 Rezultaty

Przykładowy rezultat działania algorytmu :



Losowo rozrzucone nietoperze na przestrzeni rozwiązań

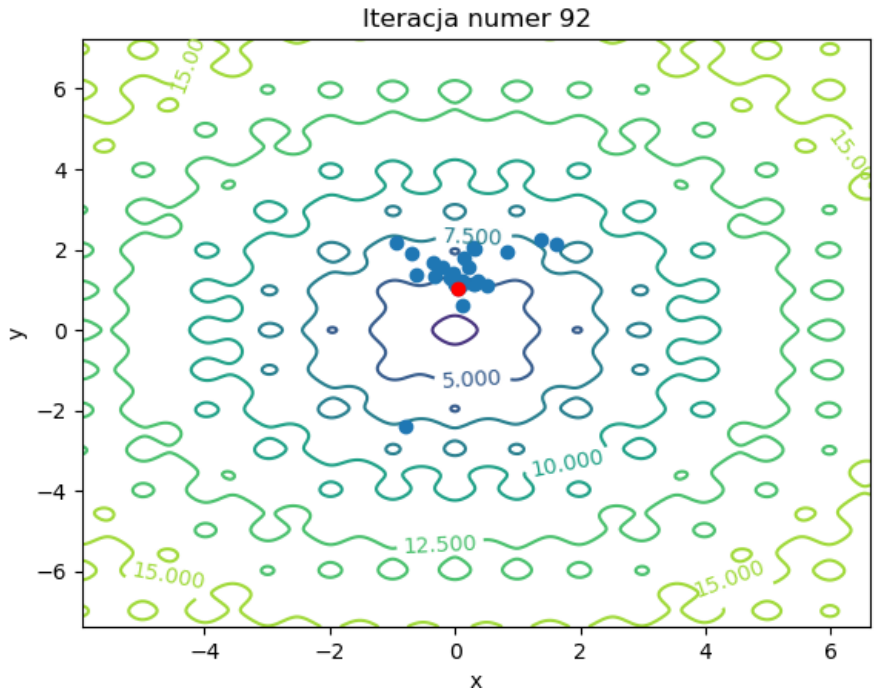


W iteracji 10 mamy wizualizacje sytuacji w której nietoperz omija ‘ścianę’ zakresu rozwiązań. Aby przeciwdziałać tej sytuacji program musi posiadać mechanizm, która pozwoli nietoperzowi skutecznie powrócić w obszar rozwiązań. Podejście do tworzenia ścian, zależy od rozwiązywanego problemu. Dla funkcji Ackley’a wykorzystujemy jej symetryczną

własność (ekstremum globalne znajduje się w centrum). Po przekroczeniu granicy przestrzeni następuje odbicie w przeciwnym kierunku (na zasadzie zderzenia sprężystego)

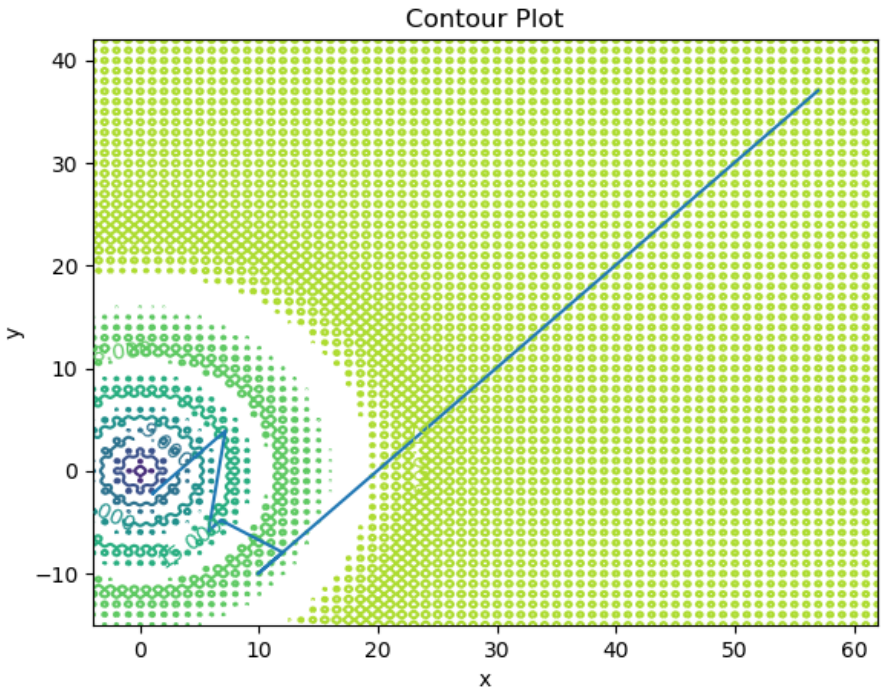
1. Ustawia nietoperza na wartości granicznej względem przekroczonej zmiennej
2. Nadajemy mu nową prędkość o przeciwnym zwrocie dla zmiennej przekraczającej, a dla drugiej wektor o tej samej długości i losowym zwrocie

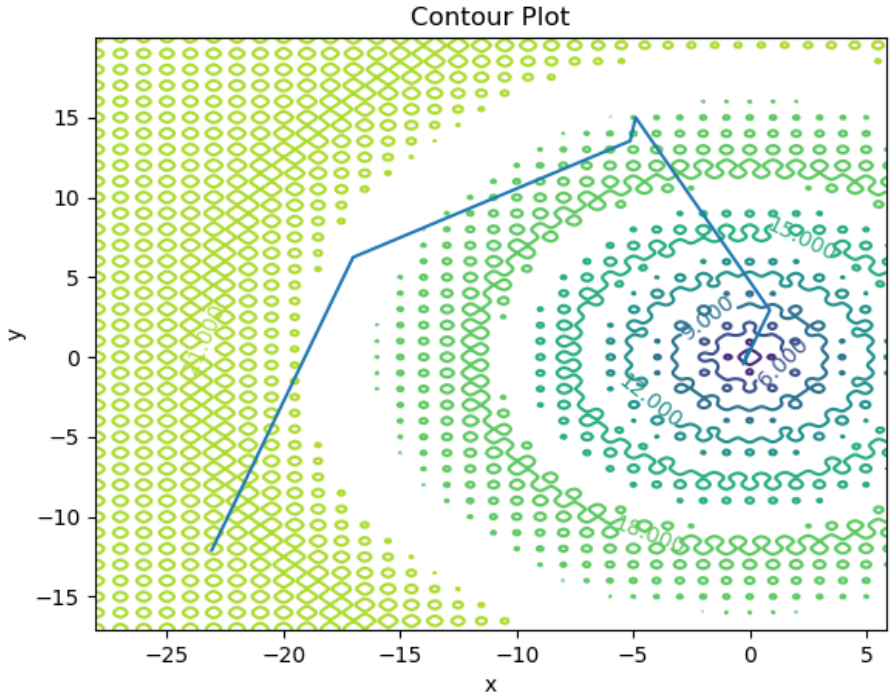
Konstrukcja funkcji Ackleya prowadzi często do sytuacji w której rozwiązanie idące w dobrym kierunku jest minimalnie gorsze od innego przedstawiciela stada. Przykładem jest wynik jednej z iteracji:

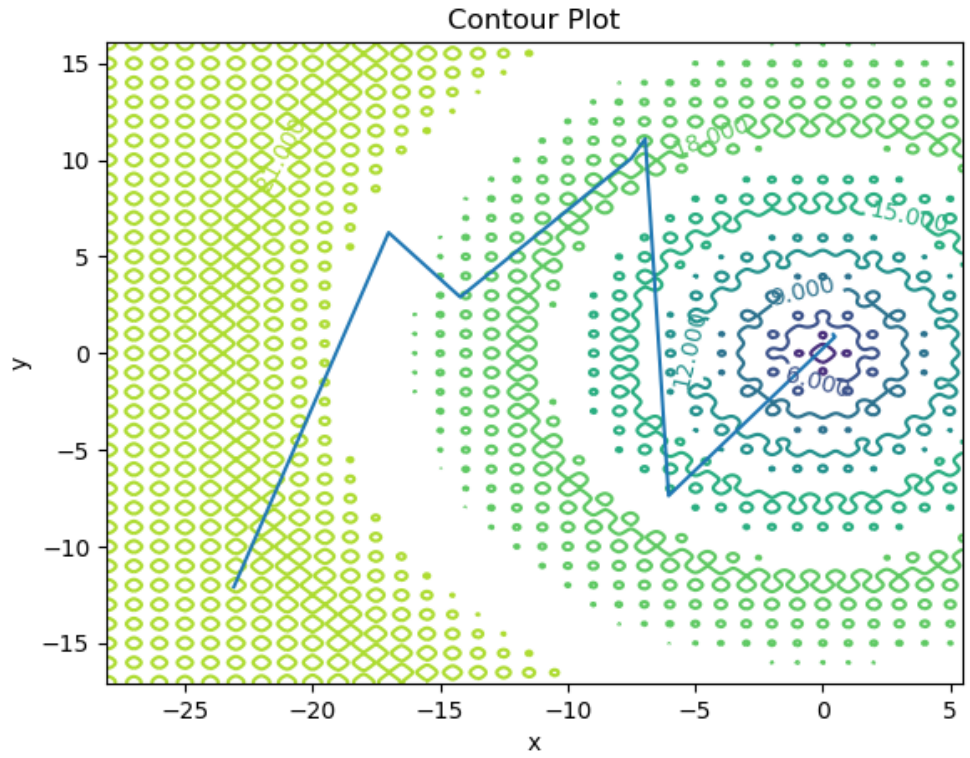


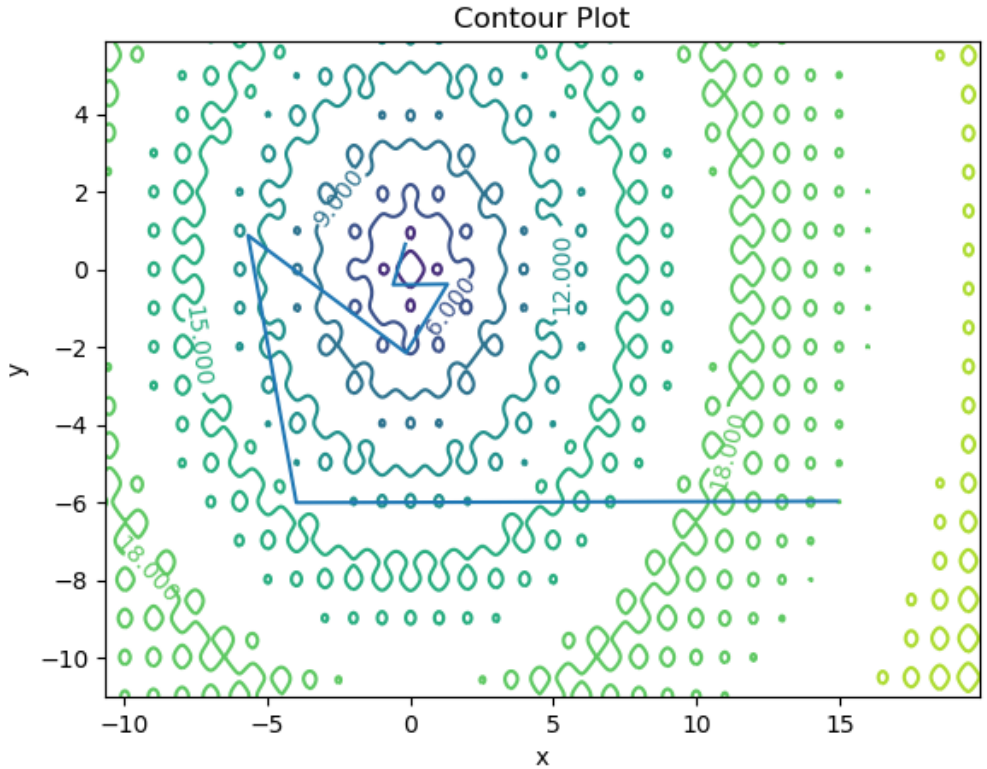
Czerwony punkt to najlepszy nietoperz w tej iteracji.

Dla każdej iteracji śledzono najlepsze położenie i konfrontowano ze stale przechowywanym i nadpisywanym najlepszym rozwiązaniem spośród wszystkich wygenerowanych:
Przykładowa wizualizacja postępu w trakcie trwania algorytmu :

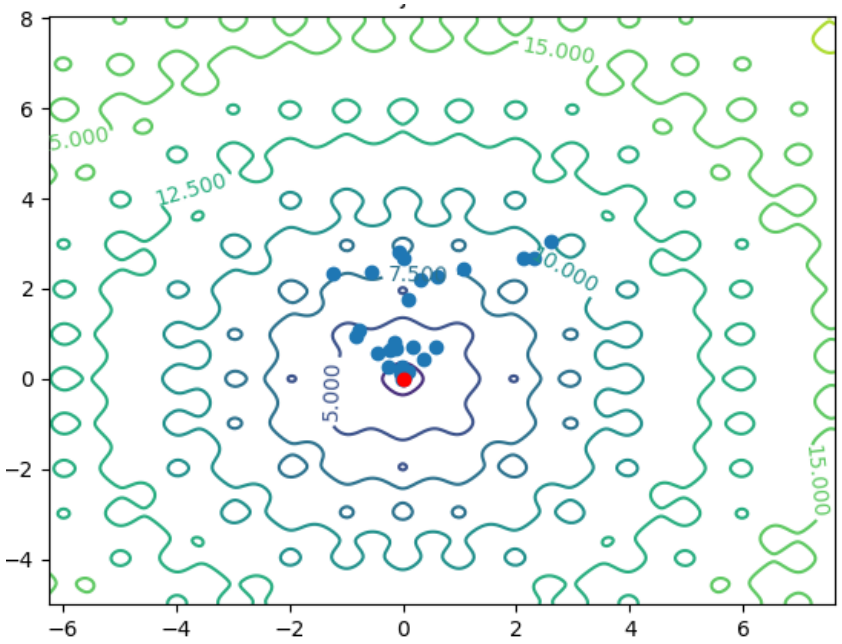

Contour Plot
40
30
20
10
0
−10
y
0
10
20
30
40
50
60
x


Contour Plot
15
10
5
0
−5
−10
−15
y
−25
−20
−15
−10
−5
0
5
x

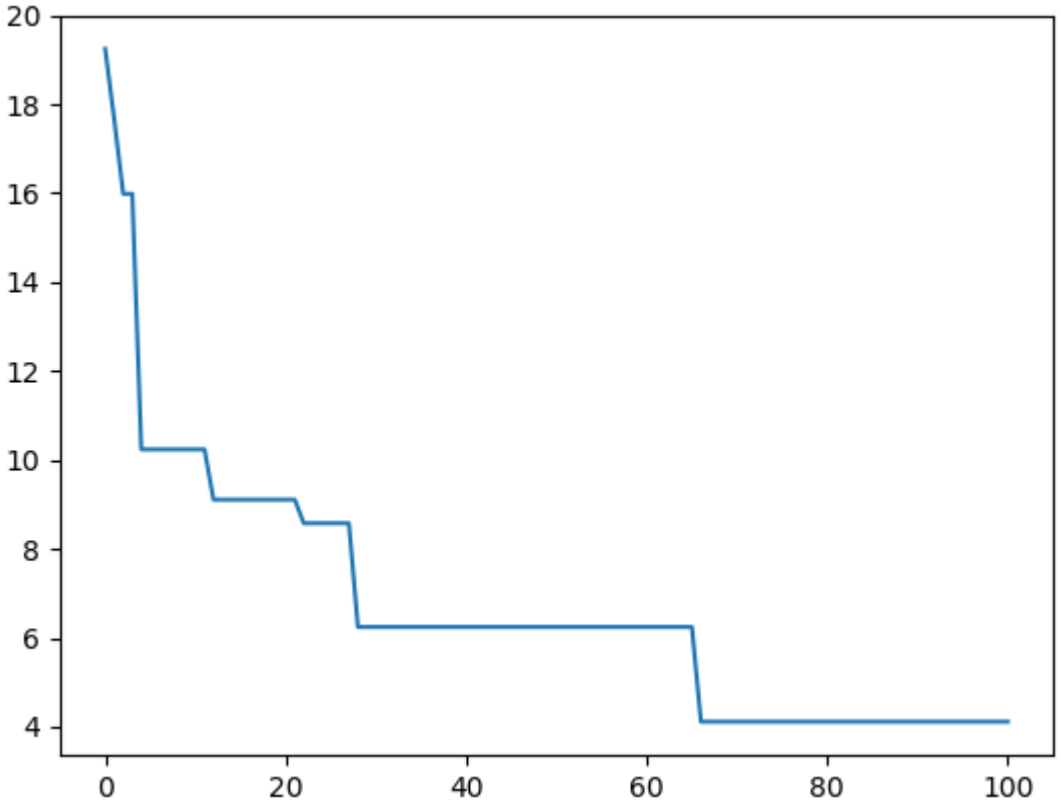

Contour Plot
15
10
5
0
−5
−10
−15
y
−25
−20
−15
−10
−5
0
5
x
21.000
18.000
15.000
9.000
12.000
6.000


Contour Plot
4
2
0
−2
−4
−6
−8
−10
y
−10
−5
0
5
10
15
x
9.000
12.000
15.000
6.000
18.000
18.000

Najlepszy zaobserwowany nietoperz trafił w ekstremum globalne:



Dodatkowo dla każdej iteracji zapisywana była informacja o położeniu najlepszego nietoperza w stadzie.



3.2 Wnioski

Algorytm nietoperza wykorzystuje zalety innych znanych algorytmów optymalizacyjnych, przeplatając i zagnieżdżając wybrane narzędzia opuszczania ekstremów lokalnych i przeszukiwania przestrzeni rozwiązań. Krytyczny wpływ na działanie algorytmu ma nie tylko dobór parametrów, ale i wzajemny balans między nimi. Równie istotny jest sposób utrzymywania stada na przestrzeni rozwiązań. Niemniej jednak metoda jest bardzo skuteczna. W przeprowadzonej symulacji dla każdego powtórzenia algorytmu wybierano kontrkandydatów w postaci losowego doboru wartości, tak aby w każdej iteracji generować tylu losowych kandydatów ile wynosiła wielkość populacji (w tym przypadku 30*201=6030). Podczas symulacji (3000 powtórzeń) dokładnie raz zdarzyła się sytuacja w której losowanie zdołało wygrać z najlepszym rozwiązaniem znalezionym przez algorytm.

4 Bibliografia

[1] Xin-She Yang ,2010, A New Metaheuristic Bat-Inspired Algorithm, University of Cambridge, UK
https://arxiv.org/abs/1004.4170

[2] Huang J.,2020, Bat Algorithm Based on an Integration Strategy and Gaussian Distribution
https://www.hindawi.com/journals/mpe/2020/9495281/