Gliwice, 08-12-2021 r.

Modelowanie matematyczne

Dokumentacja zadania laboratoryjnego - Projekt 6

Tytuł: **Proste lokalne wyszukiwanie**

Autor : Dawid Bitner

Kierunek: Informatyka, studia 2 stopnia (sem.II)

Cel zadania / projektu:

Celem zadania było przygotowanie programu implementującego algorytm prostego lokalnego wyszukiwania. Metoda ta służy do wyznaczania lokalnego ekstremum zadanej funkcji.

Opis:

Proste lokalne wyszukiwanie jest algorytmem heurystycznym co oznacza, że dla każdego wywołania procedury możliwe jest otrzymanie innego wyniku. Metody heurystyczne sprawdzają się zwłaszcza wtedy, gdy analityczne metody zawiodły lub są nieefektywne. Uzyskiwane rezultaty można polepszyć za pomocą wiedzy eksperckiej. Niestety może się zdarzyć, że metoda heurystyczna nie poradzi sobie z daną funkcją i wynik nie będzie prawidłowy, funkcja przeszukiwania lokalnego może wpaść np.: w jakieś lokalne minimum, szczególnie, jeżeli znajduje się ono blisko punktu startowego. Aby uzyskać jak najlepszy wynik zwykle procedury heurystyczne wykonuje się kilkukrotnie, a ostatecznym rozwiązaniem jest najlepszy uzyskany wynik.

Algorytm prostego lokalnego wyszukiwania:

1. Wybieramy punkt początkowy $P_1 = (x_1, y_1)$
2. W otoczeniu punktu $P_{i-1} = (x_{i-1}, y_{i-1})$ wybieramy nowy punkt taki że: $P_i = (x_i, y_i)$, gdzie$x_i \in (x_{i-1} - r, x_{i-1} + r), y_i \in (y_{i-1} - r, y_{i-1} + r)$, r-promień otoczenia
3. Jeżeli $f(P_i) > f(P_{i-1})$, wówczas zastępujemy punkt P_i punktem P_{i-1}
4. Wracamy do punktu 2.

Została stworzona funkcja search[f_, xDomain_, yDomain_, r_, iterations_], która przyjmuje następujące parametry:

- f - funkcja dwóch zmiennych, której minimum poszukujemy

- xDomain - dziedzina argumentu x jako {a,b}

- yDomain - dziedzina argumentu y jako {c,d}

- r - promień otoczenia

- iterations - iteracje

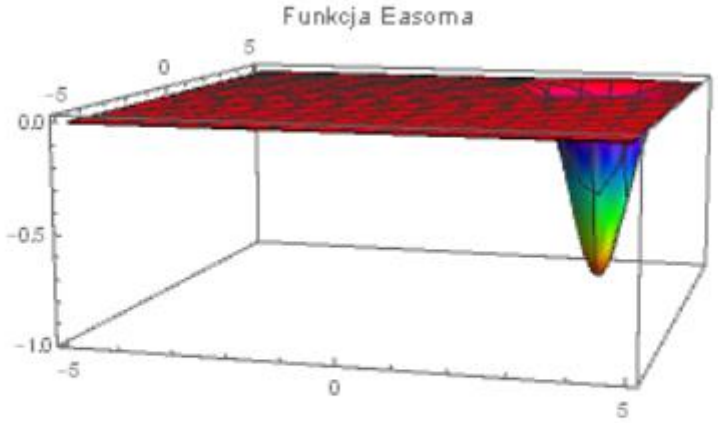
Działanie programu testowano przy wykorzystaniu funkcji Easoma, która wyraża się następującym wzorem:

$$f(x,y) = -\cos(x)\cos(y)e^{-((x-\pi)^2+(y-\pi)^2)}$$

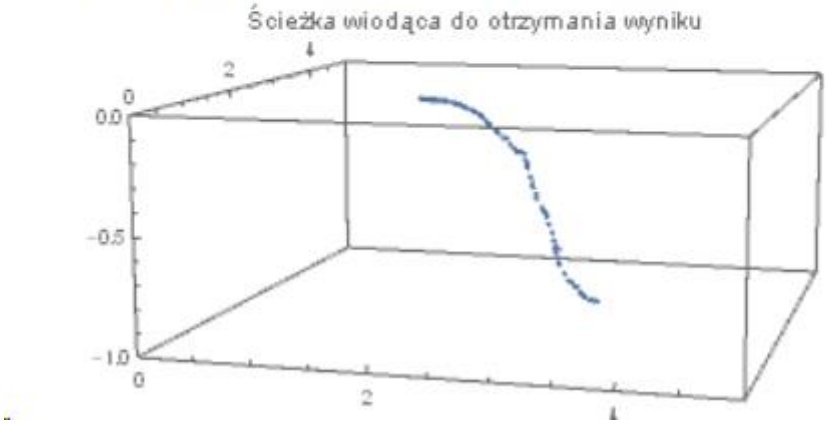
Postanowiłem użyć tej funkcji, ponieważ zdążyłem ją już przetestować na przedmiocie Heurystyczne metody optymalizacyjne.

Przykładowy wynik działania programu:

```
search[f, {0, 5}, {0, 5}, 0.05, 20000];
```




```
1 - iteracja, x = 3.14168, y = 3.14191, f(x,y) = -1.
2 - iteracja, x = 3.14165, y = 3.14168, f(x,y) = -1.
3 - iteracja, x = 3.14213, y = 3.14157, f(x,y) = -1.
4 - iteracja, x = 3.14194, y = 3.14196, f(x,y) = -1.
5 - iteracja, x = 1.30468, y = 1.30458, f(x,y) = -0.0000811021
6 - iteracja, x = 1.3053, y = 1.30507, f(x,y) = -0.0000811022
7 - iteracja, x = 3.14166, y = 3.14193, f(x,y) = -1.
8 - iteracja, x = 3.14098, y = 3.14125, f(x,y) = -0.999999
9 - iteracja, x = 1.30451, y = 1.30498, f(x,y) = -0.0000811021
10 - iteracja, x = 1.30421, y = 4.97859, f(x,y) = -0.0000811017
Znalezione minimum funkcji: f( 3.14165 , 3.14168 ) = -1.
```



Załącznik:

- Plik z programem (Dawid_Bitner_proj_6.nb)