

**Dokumentacja projektu**  
**Algorytmy i struktury danych**  
Politechnika Śląska  
Wydział matematyki stosowanej

Piotr Sobieszczyk, gr. 3I

21 maja 2020

# Część I

## Treść zadania

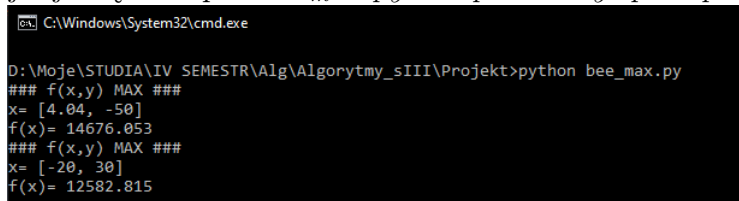
Zadaniem projektu było zmaksymalizowanie funkcji

$$f(x, y) = (\sin[x^2] + \cos[y^2] - 10)^2 + 5(x - 1)^2$$

w zbiorze  $[-50, 50]^2$  przy pomocy Algorytmu Pszczelego. Algorytm jak i przedstawienie jego działania zostało w całości zaimplementowane w języku Python.

## Instrukcja obsługi

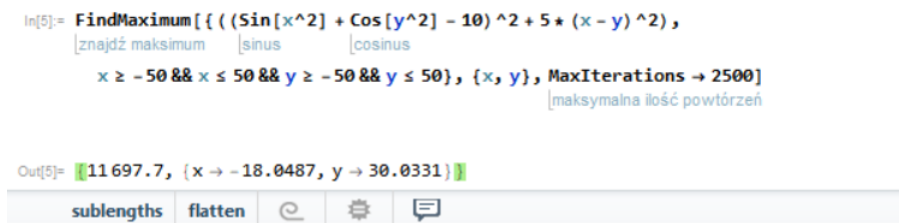
Aby uruchomić program należy uruchomić wiersz poleceń (cmd.exe) w katalogu w którym znajduje się nasz plik `bee_max.py`. Należy wpisać polecenie `python bee_max.py`.



```
C:\Windows\System32\cmd.exe

D:\Moje\STUDIA\IV SEMESTR\Alg\Algorytmy_sIII\Projekt>python bee_max.py
### f(x,y) MAX ###
x= [4.04, -50]
f(x)= 14676.053
### f(x,y) MAX ###
x= [-20, 30]
f(x)= 12582.815
```

Polecenie to wyświetli nam wynik naszego zadania. Czyli maksimum funkcji osiągnięte przez powyżej opisany algorytm. W następnym wyniku znajduje się rzeczywiste maksimum obliczone przy pomocy programu WOLFRAM MATHEMATICA.



```
In[5]:= FindMaximum[ ((Sin[x^2] + Cos[y^2] - 10)^2 + 5*(x - y)^2),
  {x, y}, {x, y}, MaxIterations -> 2500]

Out[5]:= {11697.7, {x -> -18.0487, y -> 30.0331}}
```

## Część II

### Część techniczna

Algorytm pszczeli (ang. BeA) umożliwia przeszukiwanie przestrzeni rozwiązań problemu wyrażonego funkcją kryterialną ( $F_c[U+200A]$ ) w sposób naśladujący zdobywanie nektaru przez rój pszczół. Prace nad symulacją roju pszczół były prowadzone już na przełomie lat 70. i 80. XX wieku. Obecnie istnieje duża liczba modyfikacji i rozszerzeń podstawowej wersji tego algorytmu, w tym np., zaprezentowany w dalszej części rozdziału, algorytm sztucznej kolonii pszczół ABC. W algorytmie BeA, początkowa grupa pszczół zwiadowców (Nsb) zostaje wysłana w celu przeszukania w sposób losowy obszaru i odnalezienia miejsc zasobnych w kwiaty. Po powrocie do ula pszczoły, podczas tańca pszczelego, przekazują informację innym pszczołom o swoim najlepszym odkryciu. Podczas wykonywania wywijanego tańca pszczelego (waggle dance) przekazana zostaje informacja o: lokalizacji znalezionego źródła pożywienia, czyli kierunku oraz odległości od ula oraz jakości źródła (nektaru). Do najlepszych miejsc (źródeł pożywienia) zostają wysłane pozostałe pszczoły i to one rozpoczynają zbiór nektaru. Im źródło nektaru jest zasobniejsze, tym więcej pszczół o nim się dowiaduje.

Algorytm pszczeli składa się z procedury inicjalizacji i głównym cyklu wyszukiwania iteracyjnego, które jest dla określonej liczby  $T$  razy lub dopóki roztwór dopuszczalnej sprawności znajduje. Każdy cykl wyszukiwania składa się z pięciu procedur: rekrutacja, wyszukiwanie lokalne, okolica kurczy strona zaniechanie, i globalne wyszukiwanie.

#### Proces algorytmu

**Etap 1** Losowa inicjalizacja rozwiązań początkowych – pszczoł zwiadowców

**Etap 2** Obliczenie wartości funkcji kryterialnej rozwiązań początkowych dla całej populacji,

**Etap 3** Dopóki niespełnione jest kryterium stopu (zadana maksymalna liczba iteracji) to należy:

- wybrać spośród Nsb odwiedzonych miejsc,
- zrekrutować pszczoły do najlepszych miejsc,
- wyznaczyć wartość funkcji celu,
- wybrać najlepszą pszczołę w danym miejscu (najlepsze lokalne rozwiązanie),
- przypisać pozostałe pszczoły do losowych poszukiwań,
- obliczyć dla każdej z pszczół wartość funkcji dopasowania.

**Etap 4** Jeżeli kryterium stopu zostanie spełnione (liczba iteracji osiągnięta), to wybierane jest najlepsze rozwiązanie

## Matematyczna prezentacja

Wymiana informacji między pszczołami jest modelowana równaniem:

$$p(x_i) = \frac{F(x_i)}{\sum_{i=0}^n F(x_i)}$$

gdzie  $F(x_i)$  jest wartością funkcji kryterialnej dla najlepszej pozycji zapamiętanej przez daną pszczołę. Dzięki tej informacji pszczoły lecą poszukiwać pożywienia w najlepszym kierunku. Model ruchu jest opisany równaniem:

$$x_i^{t+1} = x_i^t + \alpha_k \cdot \beta \cdot \Delta x_{ik}$$

gdzie  $k$  jest losowym indeksem pszczoły spośród najlepszych pozycji,  $\alpha_k$  jest wartością losową z przedziału  $[-1,1]$ ,  $\Delta x_{ik}$  jest obliczana

$$\Delta x_{ik} = (x_{ij} - x + kj)$$

gdzie  $j$  jest losową współrzędną

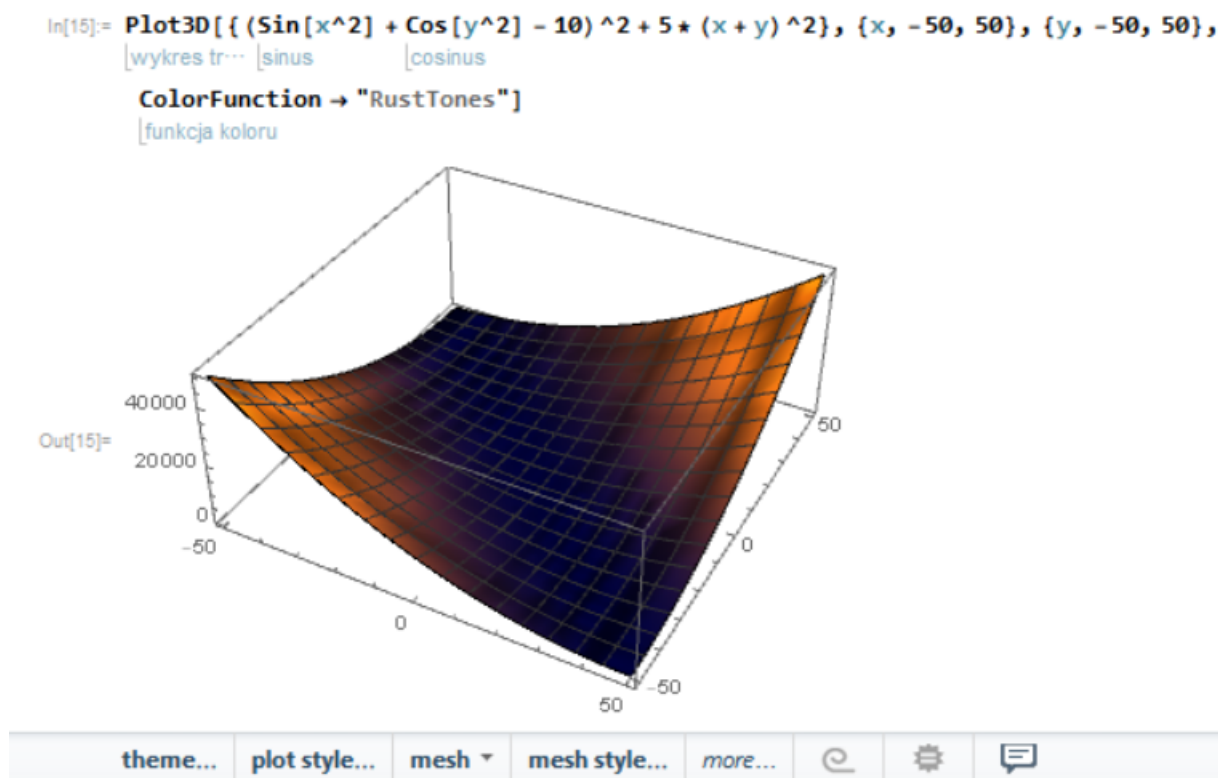
## Implementacja

Poniżej implementacja w formie pseudokodu

```
for i = 0 to Liczbageneracji do
  zwiadowca[i] = Kandydat_poczatkowy
  losowa_poszczola = obecna
  for j = 0 to LLiczbapopulacji do
    =randomfloat[-1, 1]
    k = randomint[0, 1]
  end
  Jezelifunkcja(pozycjazwiadowcy) > funkcja(pozycjaposzczoly)pozycja
  poszczyl = pozycja kandydata
end
```

## Prezentacja wynikow

Graficzna prezentacja funkcji w programie WOLFRAM MATHEMATICA



## Kod programu

- bee\_max.py

```
import random
from math import sin, cos

# funkcje
def random_(left, right, n):
    return [random.uniform(left, right) for x in range(n)]

# funkcja z zadania
def f(values):
    return ((float(sin(values[0])**2)+cos(float(values[1])**2)-10)**2 + 5*(values[0]-values[1])**2)

# tutaj przechowyjemy pozycje kazdej pszczoły
class Bee(object):
    def __init__(self, position):
        self.position = position

def BeeAlgorithm(f, left, right, n):
    generationCount = 100 # liczba generacji
    populationCount = 50 # liczba pszczoł [U+FFFD]+FFFD]

    # początkowa populacja
    population = [Bee(random_(left, right, n)) for i in range(
        populationCount)]

    for generation in range(generationCount): # pszczoły
        # pracujące
        for bee in population:
            for i in range(20): # wielokrotne szukanie przed
                # zakończeniem, w naszym przypadku 20 razy
                alfa = random.uniform(-1, 1) # losowa alfa z
                # zakresu [-1,1]

                candidatePosition = bee.position[:]

                randomBee = Bee(bee.position)

                while randomBee.position == bee.position:
                    randomBee = random.choice(population)
```

```

k = random.randint(0, 1) # zmienna 'x' albo 'y'
    ' mozliwa

candidatePosition[k] += alfa *(bee.position[k]
    - randomBee.position[k])

best = max(min(candidatePosition[k], right),
    left)
candidatePosition[k] = best

if f(candidatePosition) > f(bee.position):
    bee.position = candidatePosition[:]

# sortowanie i wybranie najlepszej pozycji
population.sort(key=lambda x: f(x.position))
#population.reverse()
population = population[: (populationCount//2)]
# nowe jedzenie
population.extend([Bee(random_(left, right, n)) for i
    in range(populationCount - len(population) + 1)])

# rozwiazanie
bestVector = population[0].position
for x in range(len(bestVector)):
    bestVector[x] = round(bestVector[x], 2)

print(f"### f(x,y) MAX ###\nx={bestVector}\nf(x)={round(f
    (bestVector), 3)}") #wynik
print(f"### f(x,y) MAX ###\nx={[-20, 30]}\nf(x)={round(f
    ([-20, 30]), 3)}") #wynik

```

```

BeeAlgorithm(f, -50, 50, 2)

```